REMARKS

In this Amendment, Applicant has cancelled Claims 27 – 28 and 32 without prejudice or disclaimer, amended Claims 3, 4 and 22, and added new Claims 33 – 42 to overcome the rejections and specify the embodiments of the present invention. It is respectfully submitted that no new matter has been introduced by the amendment. All claims are now present for examination and favorable reconsideration is respectfully requested in view of the preceding amendments and the following comments.

PRIORITY:

The certified copy of priority document has been submitted on April 11, 2008. Therefore, withdrawal of the objection is respectfully requested.

OBJECTION TO SPECIFICATION & REJECTIONS UNDER 35 U.S.C. § 112 FIRST PARAGRAPH::

The specification has been objected as failing to provide proper antecedent basis for the claimed subject matter. Claims 3, 4, 22, 27 and 28 have been rejected under 35 U.S.C. § 112, second paragraph, as allegedly failing to comply with written description requirement.

It is respectfully submitted that the currently presented claims have antecedent basis in the specification. More specifically, in Claim 3,

- "subsystems" has been deleted as it was used only in the definition of "first subsystem" and "second subsystem".
- "first subsystem" is replaced with "user control program circuit".
- "second subsystem" is replaced with "monitoring services and control unit".
- "user program circuit", as appeared previously in the claims is now "user control program circuit" as used in the specification, which is simply a fuller form name for the same entity.

- "dual-purpose flip-flop" is replaced by "logic processing circuit flip-flop".
- "shift chain" is replaced by "shift register".
- "clock to said dual-purpose flip-flops is enabled" is replaced by "clock to said logic processing circuit flip-flops is enabled".
- "pause mode in which the clock to said dual-purpose flip-flops is disabled" is replaced using the terms above.

FIG.5 and FIG. 6 illustrate the disposition of the logic processing circuit containing the user control program circuit within the invention.

The user control program circuit is a circuit designed by the user to provide the control functionality required. According to the present invention the available elements include a logic processing circuit flip-flop that has a support circuit that enables the element to act both as a flip-flop in the user control program circuit and also provide access to the state data stored in the flip-flop. Paragraphs [0208] and [0209] of the published specification support this definition.

Paragraphs generally supporting the arrangement and operation of the user control program circuit include [0080], [0105], [0122], [0130], [0132], [0181], [0202], [0209], [0217], [0220], [0237], [0256], and those adjacent paragraphs.

The logic processing circuit is generated from the user control program circuit by linking the logic processing circuit flip-flops and support circuits together, and providing the necessary control signals and clocks as described in the specification. The logic processing circuit is therefore a circuit that provides the necessary functionality both for the user control program and for data access.

The monitoring services and control unit is shown in FIG. 5 as integer 17. The monitoring services and control unit provides services required by the logic processing circuit and monitoring computer, and control and clock signals. This definition is supported in paragraph [0105].

The logic processing circuit flip-flops, as shown in FIG 9 integer 60, function both as logic elements in the user control program circuit and as part of the endless loop shift register 61. This is why these flip-flops were termed 'dual-purpose'. Paragraph [0208] provides supporting description of the arrangement of such 'dual-purpose' flip-flops. Support circuits are provided to enable such dual-purpose operation. A support circuit is shown in conjunction with a flip-flop 66 in FIG. 10. The operation of the support circuit for the flip-flop is described in paragraph [0209].

The shift register is shown in FIG. 9 and described in at least paragraph [0202].

In Claim 4, regarding the phrases "clock to said logic processing circuit flip-flops is enabled" and "pause mode in which the clock to said logic processing circuit flip-flops is disabled", the clock connection to the flip-flop is shown in FIG 10 as integer 75. FIG 7 shows a diagram of the system clocks and their timing. Disabling a clock is simply a matter of omitting the clock pulses and maintaining the clock signal at an inactive level. Enabling the clocks is a matter of providing pulses as shown. The well known method of enabling or disabling such a clock signal is to use an AND gate, preferably internal to the flip-flop and with an additional clock enable signal associated with the flip-flop, but an external gate could be used. Paragraph [0153] refers to starting, pausing resuming and stopping the user control program by control of the system clocks.

Other terms in the claim have been modified to bring them strictly into line with the terms used in the specification.

In Claim 22, the functionality claimed is broadly described in at least the 'Failure detection and correction' section of the specification.

In previous Claim 27 (now Claim 41), the functionality claimed is broadly described in the 'Speeding up program swaps' section of the specification. For example, the claim requires an incoming and outgoing LPC. These are LPC2 and LPC1 as described in the specification.

In previous Claim 28 (now Claim 40), the functionality claimed is broadly described in the 'Copying, initialising and adjusting state data during a program swap' section of the specification.

In new Claim 34, the functionality is broadly described in the 'relocation of state data' section of the specification.

In new Claim 33, the functionality is broadly described in the 'relocation of state data' section of the specification.

In addition, these claims have been amended to reflect antecedent terms used in claim 3. New claims 35 to 39 are a copy of claims 4, 22, 34, 28 and 27 and are dependant from new claim 33.

In light of the above, applicant contends that all terms used in the claims have a clear and unambiguous meaning and are fully supported by the description. Therefore, withdrawal of the objection is respectfully requested.

REJECTIONS UNDER 35 U.S.C. § 112 SECOND PARAGRAPH:

Claims 3, 4, 22, 27 and 28 have been rejected under 35 U.S.C. § 112, second paragraph, as allegedly failing to being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

It is respectfully submitted that the currently presented amendments clearly point out and define the embodiment of the present invention. More specifically, Examiner has objected to the phrase "configured including" as being fragmented and/or unclear. Applicant respectfully submits that the phrase "configured including" uses the word "configured" in the industry standard way. Use of "configured" and words related to it such as "configuration" would be clearly understood by those skilled in the art. For example, use of "configured" is common in publications of such companies as Xilinx and Altera when referring to programmable logic devices (PLDs) and FPGAs.

The word "configuration" refers to the process of writing a bit pattern into the PLD in order to cause the PLD to operate as if it contained a specific application circuit.

However, Applicant has amended Claim 3 for clarity to read "the programmable logic hardware, configured with the logic processing circuit, includes", thereby conveying a clear meaning to those skilled in the art. The term "includes" being standard claim terminology.

Examiner has objected to the limitation of "the monitoring computer" in claim 4. Applicant has removed the limitation "the monitoring computer" from claim 4.

Therefore, the rejection under 35 U.S.C. § 112, second paragraph, has been overcome. Accordingly, withdrawal of the rejections under 35 U.S.C. § 112, second paragraph, is respectfully requested.

REJECTIONS UNDER 35 U.S.C. § 102:

Claims 3, 4, and 22 have been rejected under 35 U.S.C. § 102 (b) as allegedly being anticipated by DS022 Virtex™-E 1.8 V Field Programmable Gate Arrays (v1.7) September 20, 2000, hereinafter Virtex-E.

Applicant traverses the rejection and respectfully submits that the present-claimed invention is not anticipated by the cited references. More specifically, Examiner quotes "*For in-circuit debugging, an optional download and readback cable is available. This cable connects the FPGA to a target system or PC or workstation. After downloading the design into the FPGA, the designer can single-step the logic, readback the contents of the flip-flops, and so observe internal logic state. Simple modifications can be downloaded into the system in a matter of minutes.*" and "*The TAP also supports two internal scan chains and configuration/readback of the device.*" as evidence of the disclosure of dual-purpose flip-flops in Virtex-E.

Amended Claim 3 requires:
"... a plurality of logic processing circuit flip-flops for storing state data, and for each of these flip-flops, an associated support circuit, the support circuits arranged to operate selectively between first and second conditions, wherein in said first condition the support circuits connect the flip-flops as a shift register for transporting state data into and out of the logic processing circuit, and wherein in the second condition the support circuits connect the flip-flops as logic elements of said user control program, the support circuits being selected to operate in only one of either the first condition or the second condition at any one time."

Amended Claim 3 has essentially replaced the term 'dual-purpose flip-flops' with language supported by the description to read: *"logic processing circuit flip-flops ... and ... an associated support circuit, the support circuits arranged to operate selectively between first and second conditions ..."*.

Amended Claim 3 further requires the same flip-flop to be used as part of the logic processing circuit and also as part of the shift register for reading and writing state data from the circuit. The passage the examiner has quoted from Virtex-E does not disclose the features required by amended claim 3. Further, neither the remainder of the text in Virtex-E, nor any other cited Xilinx document, nor the documents referenced in cited documents disclose the required features of claim 3.

Applicant cites the following passages supporting this argument. Virtex-E states:
Page 13 - Design Verification
"For in-circuit debugging, an optional download and readback cable is available. This cable connects the FPGA in the target system to a PC or workstation. After downloading the design into the FPGA, the designer can single-step the logic, readback the contents of the flip-flops, and so observe the internal logic state. Simple modifications can be downloaded into the system in a matter of minutes."

Page 19 - Readback

"The configuration data stored in the Virtex-E configuration memory can be read back for verification. Along with the configuration data it is possible to readback the contents all flip-flops/latches, LUT RAMs, and block RAMs. This capability is used for real-time debugging. For more detailed information, see application note XAPP138 "Virtex FPGA Series Configuration and Readback"."

Applicant therefore refers Examiner to the referenced document: XAPP138 Virtex FPGA Series Configuration and Readback (v2.7) July 11, 2002

Applicant notes that this document, although dated 2002, contains a Revisions section which indicates no applicable revisions were made between 2000 and 2002.

Applicable text from this document includes:
Page 20- *The Standard Bitstream*
"Virtex devices have the ability to be only partially re-configured or read back; however, this topic is beyond the scope of this note. For more information on partial configuration, refer to application note XAPP151 "Virtex Configuration Architecture User Guide"."

Page 24 - *Readback*
"Readback is the process of reading all the data in the internal configuration memory. This can be used to verify that the current configuration data is correct and to read the current state of all internal CLB and IOB registers as well as the current LUT RAM and block RAM values.

"Readback is only available through the SelectMAP and Boundary Scan interfaces. This application note only demonstrates the use of the SelectMAP interface for performing readback. For information on using the Boundary Scan interface for readback refer to application note XAPP139 "Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary Scan".

Readback Verification and Capture

Readback verification is used to verify the validity of the stored configuration data. This is most commonly used in space-based applications where exposure to radiation may alter the data stored in the configuration memory cells.
Readback capture is used to list the states of all the internal flip-flops. This can be used for hardware debugging and functional verification. When Capture is initiated, the internal register states are loaded into unused spaces in the configuration memory which may be extracted after a readback of the configuration memory.

The CAPTURE_VIRTEX component is used in the FPGA design to control when the logic states of all the registers are captured into configuration memory. The CLK pin may be driven by any clock source that would synchronize Capture to the changing logic states of the registers. The CAP pin is an enable control. When CAP is asserted, the register states are captured in memory on the CLK rising edge."

Page 33 - *Capturing Register States*
If Capture has been used to include the states of all internal registers in the readback data stream, the logic allocation <design>.ll file can be used to locate those signal state bits within the data frames. The logic allocation file includes all CLB and IOB outputs as well as all block RAM values, whether they are used in the design or not.
An example of portions of a Logic Allocation file is shown in the following text. Each Bit line includes a <Bit offset> <Frame> <Frame offset> <CLB_location.Slice> <Type> and a user net name if this node is used in the design.

Therefore, Virtex-E uses a technique known as 'Boundary Scan' to facilitate reading data from internal memory. Virtex-E does not describe any other way state data can be accessed.

Applicant therefore refers Examiner to the referenced document:
XAPP139 - Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary Scan. This document provides details about using Boundary Scan methods to access user circuit data. The document contains no relevant information not present in the other documents discussed, other than a sample implementation of a user data register shown in

Figure 3, which clearly uses Boundary Scan methods. This figure shows "cap" signals, presumably signals to be 'captured' into configuration memory by the CAPTURE_VIRTEX component.

For clarification on what 'Boundary scan' is and how it operates the Applicant refers the examiner to Texas Instruments IEEE 1149.1 Primer. This document is freely available via the Texas Instruments website: http://focus.ti.com/lit/an/ssya002c/ssya002c.pdf. A copy of this document is enclosed with this response.

IEEE 1149.1 is an introduction to Boundary Scan technology and is therefore useful to illustrate the kind of scan chains that are used by Xilinx Virtex-E. IEEE 1149.1 mandates two data registers: boundary scan and bypass. The document does provide for additional Design-Specific Test Data Registers. Xilinx calls these 'user defined registers'. However, the document does not define what they are to consist of, and leaves that to the designer. The flip-flops in the mandated registers that do the data transportation (shifting) are not part of the application or device function specific circuits. Instead, the registers are additional and are provided for the purpose of sampling device response signals for test purposes, transporting the samples out of the device for interpretation, transporting bit values into the device and providing those bit values to device functional circuits for use as test stimuli.

The registers are illustrated in IEEE 1149.1 Figures 3-6 and with more detail in Figures 3-7. The Xilinx specific equivalent can be seen in Xilinx Virtex™-E 1.8 V Field Programmable Gate Arrays, DS022 (v1.7) September 20, 2000 Figure 11. Further, on page 12, Xilinx describe the mandatory registers and how to connect user defined registers to the built-in JTAG controller and system. Xilinx do not define the user defined registers, although 'samples' are shown in XAPP139 Figure 3.

With Boundary Scan as described in IEEE 1149.1 and Virtex-E, user application circuit signals of interest (for example these may be flip-flop output signals) are provided to separate flip-flops in the scan chain. Virtex-E has a scan chain which serves as a configuration memory.

To determine the logic level of a signal in the circuit under test, a flip-flop in the scan chain (i.e., a flip-flop that is separate from any flip-flops in the circuit under test) samples and latches the logic level to allow it to be shifted out of the circuit along the scan chain to a monitoring device. In Virtex-E, sampling is performed by what Xilinx term "unused spaces" in the configuration memory scan chain.

Similarly, with Boundary Scan, values from the monitoring device may be shifted into the scan chain. When the value is established in the scan chain flip-flop associated with the desired, but separate, application circuit flip-flop, the value may be latched into that application circuit flip-flop.

Amended Claim 3 of the present application requires *"logic processing circuit flip-flops ... and ... an associated support circuit, the support circuits arranged to operate selectively between first and second conditions".* Amended claim 3 therefore requires that a single flip-flop is used to transport the data via shifting (i.e. the *"support circuits operated in the first condition connecting the flip-flops as a shift register"*) and to provide flip-flop functionality for the user's application circuit operation (i.e. the *support circuits operated in the second condition connecting the flip-flops as logic elements of said user control program circuit"*).

The circuits and literature cited by the examiner all rely on an extra flip-flop in a separate scan chain to perform the access by shift register. Neither the flip-flops in the user application circuit, nor those in the scan chain operate both as part of the shift register and also as part of the user program circuit as required by amended claim 3.

The invention recited in amended Claim 3 is shown to reduce the number of flip-flops required, as well as making access to the data of interest more flexible, easier and quicker. The distinct speed advantages have been pointed out to Examiner in previous correspondence.

Therefore, the presented claims are not anticipated by Virtex-E and the rejection under 35 U.S.C. § 102 (b) has been overcome. Accordingly, withdrawal of the rejections under 35 U.S.C. § 102 (b) is respectfully requested.

<u>REJECTIONS UNDER 35 U.S.C. § 103</u>:

Claims 27 – 28 have been rejected under 35 U.S.C. § 103 (a), as allegedly being obvious and unpatentable over Virtex-E in view of New (US 6,091,263).

Applicant traverses the rejection. It is respectfully submitted that the cited references fail to render the embodiments of the present invention as claim obvious. As explained above, the cited references are significantly different from the present invention as defined in the new Claims 40 – 41 (corresponding to previous Claim 28, 27). As Virtex-E does not disclose the required features of new Claims 40 – 41, this rejection is moot. However, New et al does not disclose 'means to support relocation of state data during a program swap operation'.

Examiner has not construed the meaning of the words 'relocate' or 'relocation' in a way consistent with the specification. Examiner is directed to the 'Relocation of state data' section of the specification for clarification (Paragraphs [0247] ... [0257]).

Applicant does not use the term 'relocate', at least in association with program swapping and associated state data, to mean simply moving something from one place to another. Specifically, Applicant's use of "relocate" and "relocation" refers to the process whereby, during a program swap operation, we move state data saved from the outgoing user program circuit into the incoming user program circuit but most probably at a different address from that at which it was read.

During this process, some data will be transferred relocated, some will be discarded, and some newly included data will be initialised. The overall operations performed on the data-set to be shifted into an incoming circuit shift register therefore include re-ordering some existing bits, omitting some existing bits, and including some new bits.

This movement of data involves placing each bit of incoming data at an address in the incoming circuit that is potentially different to the address from which it was read in the outgoing circuit. This requirement arises because the incoming and outgoing circuits differ from each other even though the incoming circuit is a modification of the outgoing circuit, therefore it cannot be assumed that the bits in the incoming circuit that have corresponding bits in the outgoing circuit will be located at the same addresses by the configuration bit-stream generating process. However, the access addresses for both the incoming and outgoing cases will be known to the monitoring computer after the configuration bit-streams have been compiled, and therefore the necessary cross-reference information is known and the relocation address memory 87 can be loaded with the cross-reference information.

The relocation process as claimed is therefore not just a simple transfer as is taught in New et al. Instead, each state data bit is examined, and if the same bit exists in both the outgoing and incoming user program circuits, then the saved state data value is written into the corresponding bit in the incoming user program circuit.

Applicant has fully considered the disclosure of New et al and finds there are three mechanisms that may be considered to be associated with the relocation of state data as defined in a general sense. Theses are:

1. The ability to save data from a state data bit (one of the bits exemplified by flip flops 104 and 105 and function generators 101 and 102 in US 6091263) to any one of a number of cache memory locations, and to restore data to the same state data bit from any one of the same cache memory locations.

There is no support disclosed with this mechanism for restoring the data for a state data bit from the cache memory locations associated with any other state data bit than the state data bit from which it was saved. Therefore this mechanism does not support circuit data relocation as required by old claims 27 and 28, or new claims 34, 28 or 27.

2. The ability to use one configuration cache memory half array as a second level configuration cache memory half array for the purposes of expanding the number of RAM caches associated with each CLB in one half of the FPGA while operating the other half of the FPGA with a fixed configuration.

The mechanism is disclosed as operating by first transferring a set of configuration data in the second level configuration cache memory half array to the first level configuration cache memory half array performed in approximately 50 microseconds by using all of the buses between the CLB rows in parallel, and then completing the configuration with the data now in the first level cache. Therefore this mechanism does not support circuit data relocation as required by old claims 27 and 28, or new claims 34, 28 or 27.

3. The ability to externally read the cache memories from a position external to the system via the configuration port and associated busses. In as much as this provides random access to the cache memories such that a bit may be read from one place and written to another, then this of course provides the ability to relocate state data when the term "relocate" is used in a general sense.

There is, however, no suggestion in US 6091263 of any requirement, method, procedure or process involving relocation as used by the Applicant. The mechanism disclosed in US 6091263 is no more designed to facilitate functional relocation than is a simple random access memory. Therefore this mechanism does not support circuit data relocation as required by previous Claims 27 and 28, or new Claims 34, 40 or 41.

New et al have therefore demonstrated that state data may be used in two ways. Firstly for initialising a circuit after configuration, and secondly for passing state data generated by circuit operation to a different circuit for further processing.

New et al make no mention of (re)storing any state data value into any bit other than the same physical bit in which the value was generated. Relocation of state data into locations that are physically different but functionally equivalent is quite different to what is taught by New et al. New et al's teaching requires the data transfer interface in an incoming configuration to be physically identical to that of the outgoing configuration.

Access to state data in a programmable logic device has always been available via the use of direct connections, although such methods are comparatively inefficient. Also, the purpose and well established use of IEEE 1149.1 is to provide a standard means of read (i.e. 'readback') and write access to digital data existing in integrated circuits in general and of any type. The novelty and value in the access methods used in such devices as Virtex-E and New et al relate to how that access is achieved and what benefits accrue from the particular methods. The novelty and value in the access methods used in such devices as Virtex-E and New et al does not relate to the provision of access per se.

The invention as claimed by the Applicant provides an alternative to the methods used in devices such as Virtex-E and New et al. The invention as claimed is preferable for applications as defined in the specification as it provides inherent speed and control advantages. Application therefore contends all claims are novel and inventive. Application requests that the claim rejections are withdrawn in light of the amended claims and arguments presented.

Therefore, the rejection under 35 U.S.C. § 103 has been overcome. Accordingly, withdrawal of the rejection under 35 U.S.C. § 103 is respectfully requested.

Having overcome all outstanding grounds of rejection, the application is now in condition for allowance, and prompt action toward that end is respectfully solicited.

Respectfully submitted,

JACOBSON HOLMAN PLLC

By [illegible]
John C. Holman
Registration No. 22,769

Date: September 12, 2008
(202) 638-6666
400 Seventh Street, N.W.
Washington, D.C. 20004
Atty. Dkt. No.: P67300US0

Enclosure:

Texas Instruments IEEE 1149.1 Primer. From http://focus.ti.com/lit/an/ssya002c/ssya002c.pdf (130 pages)

TEXAS INSTRUMENTS

IEEE Std 1149.1 (JTAG) Testability

Primer

1997 Semiconductor Group

IEEE Std 1149.1 (JTAG) Testability Primer

SSYA002C

TEXAS INSTRUMENTS

TRADEMARKS

SUPPORT

Device Boundary-Scan Description Language (BSDL) files and other information regarding Texas Instruments IEEE Std 1149.1/JTAG/boundary-scan products are maintained on the World-Wide Web at URL http://www.ti.com/sc/docs/jtag/jtaghome.htm; or, on the main Texas Instruments home page at URL http://www.ti.com. Search for keywords JTAG and testability.

Contents

Illustrations

Tables

Chapter 1
Introduction

Design for test (DFT), also known as *design for testability,* is a process that incorporates rules and techniques in the design of a product to make testing easier.

Structured design for test is a system methodology rather than a collection of discrete techniques. This methodology impacts all phases of a product's life, from device circuit design through field service. Design for test is used to manage complexity, minimize development time, and reduce manufacturing costs.

Testing has two major aspects: control and observation. To test any system it is necessary to put the system into a known state, supply known input data (test data), and observe the system to see if it performs as designed and manufactured. If control or observation cannot be carried out, there is no way to know empirically if the system performs as it should.

During the normal product development flow, testing (it may be known by different names) takes place at many points during the process. If testing is considered at the chip design level, its benefits can be used at all levels of electronic assembly, from chip through system level. See Figure 1-1.

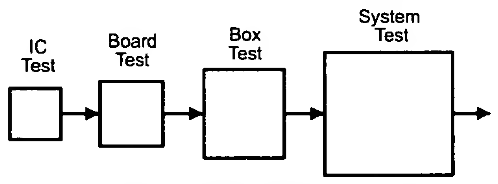


Figure 1-1. Chip Through System-Level Test

Designers usually test various functions to validate their design. Manufacturing and customer groups subject the design to an assortment of unique criteria to see if the concept works in practice. Is it manufacturable? Will it stand up to real-world operating conditions? Will repair be cost efficient? In addition to direct testability considerations, production managers want features designed into the product to help them minimize scrap and manufacturing costs. Good system-testability methodology provides an integrative function throughout the product development cycle and allows materials created during an early phase of development to be reused in later phases. Various chip designers have used this integration feature as a tool to help manage the development of complex products.

Testability provides companies with a firmer grasp on the economic and market-window constraints due to product development. One major workstation manufacturer claimed:

- Test program development would have been nearly impossible without scan techniques.
- Chip-level test development time fell from 1 man-year to about 20 hours.
- Board-level test development time fell from multiple man-years to about a week.
- Three months were cut off development time.

Overall Rationale for Design for Test

Manufacturers of state-of-the-art electronic products face a unique set of problems. Although modern circuit density, high device speed, surface-mount packaging, and complex board-interconnect technology have a positive influence on state-of-the-art electronic systems, these factors can adversely affect ability to verify correct design and operation. Increased complexity and lack of physical access to circuitry makes for costly and time-consuming testing using traditional test techniques.

Reduced Cost and Higher Quality

Reacting to this complexity, with an eye on the bottom line, manufacturers may opt to perform less rigorous testing. Manufacturers who choose the less rigorous testing as an expeditious alternative to the expense of full testing gamble their technical credibility in the marketplace and expose themselves to the high cost of product returns. In today's global electronics marketplace, a manufacturer who delivers poorly tested products does not remain competitive. The cost for detecting and identifying faults using traditional test methods increases by an order of magnitude as a circuit's level of complexity increases. These increased costs and development time reduce profit margins, delay product introduction, and reduce time-to-market windows. An increasing number of companies have simultaneously improved their product quality and profit margins by adopting system-level (integrative) design methods. Design for test is one such system-level approach.

Benefits Over Standard Test Methods

Time to market is more important than ever in the high technology marketplace. Companies that can produce quality products with a short product development cycle-time have a competitive advantage. Designing testability into a system can play an important role in introducing a new high-technology product with an expected five-year life cycle to market on time. Table 1-1 shows various product development time/budget scenarios and the resulting project profitability.

Table 1-1. High-Technology Product Scenarios†

	Product A	**Product B**	**Product C**
To Market:	on time	on time	6 mos. late
Budget:	on	50% over	on
Available Profit Over 5 Years:	100%	96%	66%

†Source:McKinsey & Company

Adding testability to a product increases design time and costs, while reducing costs of design validation, manufacturing test, and system maintenance.

The system design phase of product development represents only 15 percent of a product's total life-cycle cost. However, the system design phase has a 70-percent impact on a product's operation and support costs over the product's total life (source: Mitre Corporation, 1987 Government Microcircuit Applications Conference).

The majority of faults found on boards, such as solder joints (shorts and opens), components (wrong device, missing device, wrong orientation, wire-bond failure, and stuck pins), etch integrity, and connector faults, make up over 95 percent of failures found. A structured technique such as boundary-scan testing allows for pins-out testing to easily detect these failures (source: Teradyne).

The additional cost of designing testability into a system during the system design phase can be more than offset over the product's total life.

Design cycle times have shortened significantly over the years while test program development time has increased, necessitating that companies adopt structured or repeatable methodologies. Table 1-2 documents the increase in test program development time as test requirements increase.

Table 1-2. Time to Develop Test Programs (in Man-Months)†

1987–1980	3–6 months
1981–1983	6–12 months
1984–1986	9–18 months
1987–1990	12–24 months

†Source: Texas Instruments

Standard Test Solutions Versus Proprietary Solutions

Embedded test, emulation, and maintenance circuitry are well defined and understood within the test community. Previously, the lack of standards caused these structures to be implemented in an *ad hoc* and proprietary manner. Since proprietary solutions are usually more expensive and labor intensive, the added costs further limited the use of these test circuits. Boundary-scan testing combined with a common test bus interface and test protocol has these benefits:

- Provides a standard and cost-effective solution to traditional test problems
- Opens new applications

The ability to reuse previously developed test data and to use less costly test equipment means that this approach yields products that are less expensive to manufacture.

An Industry Standard — IEEE Std 1149.1-1990 (JTAG)

In 1985, an *ad hoc* group composed of key electronic manufacturers joined to form the Joint Test Action Group (JTAG). JTAG had over 200 members around the world, including major electronics and semiconductor manufacturers. This group met to establish a solution to the problems of board test and to promote a solution as an industry standard. The solution, which became IEEE Std 1149.1-1990, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, is the basis for Texas Instruments (TI™) testability products. IEEE Std 1149.1 allows test instructions and data to be serially loaded into a device and enables the subsequent test results to be serially read out.

Every IEEE Std 1149.1-compatible device has four additional pins — two for control and one each for input and output serial test data. To be compatible, a component must have certain basic test features, but IEEE Std 1149.1 allows designers to add test features to meet their own unique requirements. The specification was adopted as an IEEE standard in February 1990.

Chapter 2
Benefits of Testability

This chapter explains how designing testability into devices eliminates problems associated with traditional testing and improves quality and efficiency.

Traditional Testing

Traditional board-level and device-level testing consumes a great deal of time and requires special hardware and complex automatic test equipment (ATE) for each type of board or device. This results in increased costs and development time. In addition, extensive testing is necessary for the evermore stringent reliability standards and performance standards in the defense, aerospace, automotive, computer, and communications industries. These extensive tests can delay the market introduction of products, disrupt just-in-time (JIT) manufacturing flows, and limit the productivity of standard ATE operations. This creates numerous problems because time to market is more important than ever in the high-technology marketplace. Companies that produce quality products with a short product-development cycle time have a competitive advantage.

Efficient Testing

An innovative approach to the problems inherent with traditional testing is to incorporate design-for-test techniques that allow embedded testing to be performed. For example, data can be scanned in to stimulate internal system nodes while the component or circuit is embedded within the system. During the same scan, the previous condition of each node is scanned out. This saves test time and reduces the number of test vectors needed.

Lower Cost for Testing

The additional cost of designing testability into a system during the design phase is more than offset over the product's total life. This is accomplished by reducing the test program development time, minimizing fixture complexity, and allowing for the use of lower-cost ATE solutions. Another cost benefit is the economy of scale gained by having a standard test approach that spans design, test, manufacture, field repairs, and maintenance.

Production Time Savings

Board-level boundary-scan testing is easily implemented using TI's line of IEEE Std 1149.1 testability devices, such as:

- Widebus™ and octal bus interfaces
- Scan-support devices
- ASIC and DSP

These IEEE Std 1149.1-compliant devices are included in board design with little modification to existing circuitry. Embedded testability greatly reduces the need for other test points on the board, and offers these advantages:

- Greatly simplified test fixtures
- Reduced fixture construction time
- Sophisticated built-in test and debug operations

Many ICs or boards can be tested together using the serial IEEE Std 1149.1 test bus under the control of boundary-scan test software.

Easier Board-Level Isolation

Fault isolation on a printed circuit board can be greatly improved by electronically isolating suspect areas using boundary-scannable devices. The IEEE Std 1149.1 test bus controls boundary-scannable devices to place them in *EXTEST* for pins-out testing. This effectively partitions or isolates circuitry for separate testing. Partitioning a system using IEEE Std 1149.1-compliant devices reduces the number of patterns required for testing each circuit area. See Figure 2-1 for an example of a design than can be partitioned.

Simple Access to Circuits

Highly integrated, modern, multilayer systems or ICs with fine-pitch pins are virtually impossible to access using manual probes or ATE. Some boards require extensive fixturing or redesign before they can be tested effectively. TI's testability devices with boundary-scan architecture eliminate physical access problems. These parts provide the designer with testability for the most complex and hard-to-access circuits, and add controllability of test circuits. In addition, a designer can easily observe and control internal device functions.

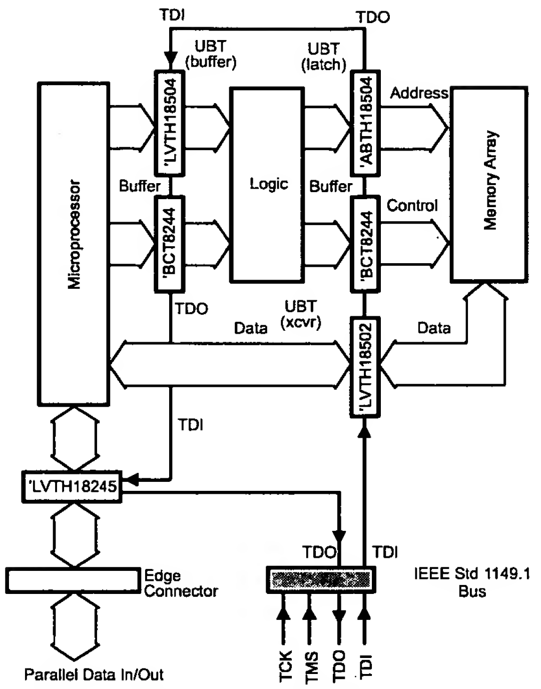


Figure 2-1. Boundary-Scan Testing Using the IEEE Std 1149.1 Bus

Chapter 3
Boundary-Scan Architecture and IEEE Std 1149.1

Boundary scan is a special type of scan path with a register added at every I/O pin on a device. Although this requires the addition of a special test latch on some pins, the technique offers several important benefits. The most obvious benefit offered by the boundary-scan technique is allowing fault isolation at the component level. Such an isolation requirement is common in telecomunications switching environments where prompt field repair is critical.

A major problem driving the development of IEEE Std 1149.1 boundary scan is the adverse effect of surface-mount technology. The inclusion of a boundary-scan path in surface-mount components, in many cases, affords the only way to perform continuity tests between devices. By placing a known value on an output buffer of one device and observing the input buffer of another interconnected device, it is easy to see if the printed wiring board (PWB) net is electrically connected. Failure of this simple test indicates broken circuit traces, cold solder joints, solder bridges, or electrostatic-discharge (ESD) induced failures in an IC buffer — all common problems on PWBs.

A less-obvious advantage of the boundary-scan methodology is the ability to apply predeveloped functional pattern sets to the I/O pins of the IC by way of the scan path. IC manufacturers and ASIC developers create functional pattern sets for DC test purposes. Subsets of these patterns can be reused for in-circuit functional IC testing. Reusing existing patterns in the development of system diagnostics can save large amounts of development resources, especially if many of the ICs in a system have embedded boundary-scan paths.

IEEE Std 1149.1 is a common protocol and boundary-scan architecture developed into an industrial standard after thousands of man hours of cooperative development by approximately 200 major international electronics firms. Early contributors in the development of IEEE Std 1149.1 were AT&T, DEC™, Ericsson, IBM™, Nixdorf, Philips, Siemens, and TI. These companies recognized that only a nonproprietary architecture would encourage companies to offer the compatible integrated circuits, test equipment, and CAD software needed to bring product development, manufacturing, and test costs under control in today's competitive electronics marketplace. Many people believe that boundary-scan architecture will do for development, manufacturing, and test what the RS-232C standard did for computer peripherals.

Boundary-Scan Overview

Boundary scan is the application of a scan path at the boundary (I/O) of ICs to provide controllability and observability access via scan operations. Figure 3-1 shows an IC with an application-logic section and related input and output, and a boundary-scan path consisting of a series of boundary-scan cells (BSCs), in this case one BSC per IC function pin.

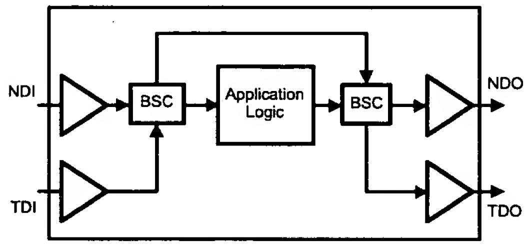


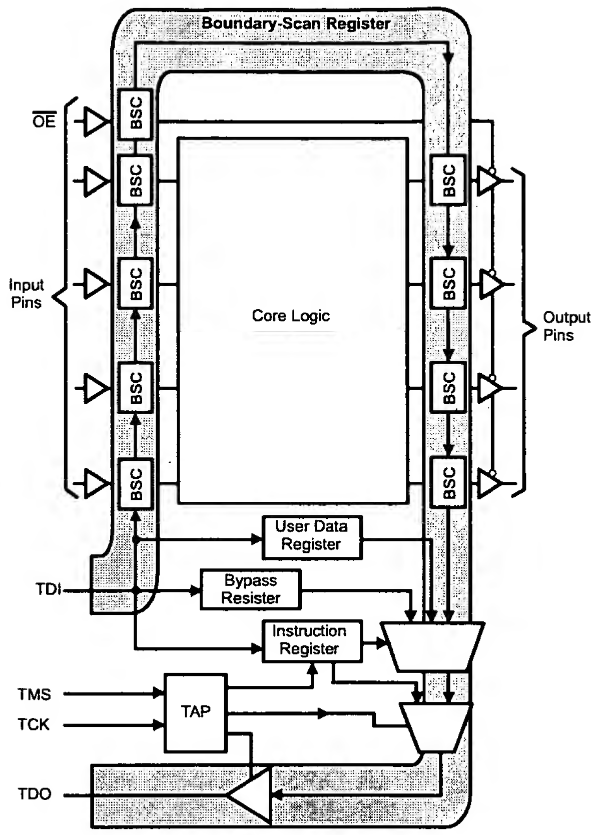
Figure 3-1. Boundary-Scan Example

The BSCs are interconnected to form a scan path between the host IC's test data input (TDI) pin and test data output (TDO) pin. During normal IC operation, input and output signals pass freely through each BSC, from the normal data input (NDI), to the normal data output (NDO). However, when the boundary-test mode is entered, the IC's boundary is controlled in such a way that test stimulus can be shifted in and applied from each BSC output (NDO), and test response can be captured at each BSC input (NDI) and shifted out for inspection. External testing of wiring interconnects and neighboring ICs on a board assembly is accomplished by applying test stimulus from the output BSCs and capturing test response at the input BSCs. As an option, internal testing of the application logic can be accomplished by applying test stimulus from the input BSCs and capturing test response at the output BSCs. The implementation of a scan path at the boundary of IC designs provides an embedded testing capability that can overcome the physical access problems in current and future board designs.

Test Interface and Boundary-Scan Architecture

Figure 3-2 shows the IEEE Std 1149.1 architecture. The architecture consists of an instruction register, a bypass register, a boundary-scan register (highlighted), optional user data register(s), and a test interface referred to as the test access port (TAP). In Figure 3-2, the boundary-scan register (BSR), a serially accessed data register made up of a series of boundary-scan cells (BSCs), is shown at the input and output boundary of the IC.

The instruction register and data registers are separate scan paths arranged between the primary test data input (TDI) pin and primary test data output (TDO) pin. This architecture allows the TAP to select and shift data through one of the two types of scan paths, instruction or data, without accessing the other scan path.

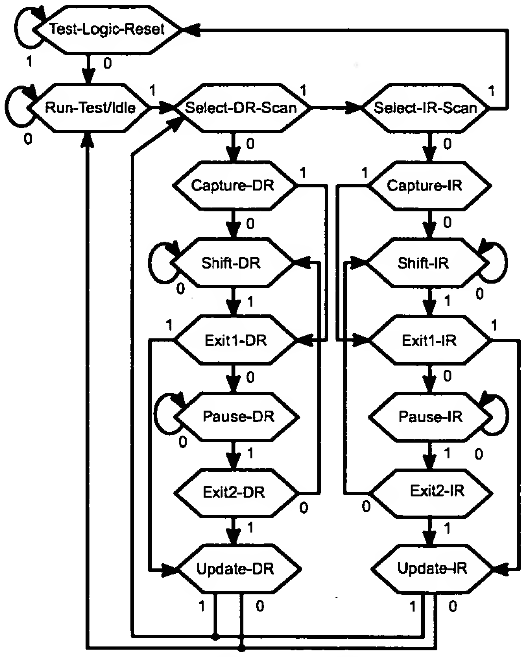


Note: The boundary-scan register is shifted TDI to TDO.

Figure 3-2. Boundary-Scan Architecture

Test Access Port and Operation

The TAP is controlled by the test clock (TCK) and test mode select (TMS) inputs. These two inputs determine whether an instruction register scan or data register scan is performed. The TAP consists of a small controller design, driven by the TCK input, which responds to the TMS input as shown in the state diagram in Figure 3-3. The IEEE Std 1149.1 test bus uses both clock edges of TCK. TMS and TDI are sampled on the rising edge of TCK, while TDO changes on the falling edge of TCK.



Note: The value shown adjacent to each state transition in this figure represents the signal present at TMS at the rising edge of TCK.

Figure 3-3. TAP Controller State Diagram

The main state diagram consists of six steady states: Test-Logic-Reset, Run-Test/Idle, Shift-DR, Pause-DR, Shift-IR, and Pause-IR. A unique feature of this protocol is that only one steady state exists for the condition when TMS is set high: the Test-Logic-Reset state. This means that a reset of the test logic can be achieved within five TCKs or less by setting the TMS input high.

At power up, or during normal operation of the host IC, the TAP is forced into the Test-Logic-Reset state by driving TMS high and applying five or more TCKs. In this state, the TAP issues a reset signal that places all test logic in a condition that does not impede normal operation of the host IC. When test access is required, a protocol is applied via the TMS and TCK inputs, causing the TAP to exit the Test-Logic-Reset state and move through the appropriate states. From the Run-Test/Idle state, an instruction register scan or a data register scan can be issued to transition the TAP through the appropriate states shown in Figure 3-3.

The states of the data register scan and instruction register scan blocks are mirror images of each other, adding symmetry to the protocol sequences. The first action that occurs when either block is entered is a capture operation. For the data registers, the Capture-DR state is used to capture (or parallel load) the data into the selected serial data path. If the BSR is the selected data register, the normal data inputs (NDIs) are captured during this state. In the instruction register, the Capture-IR state is used to capture status information into the instruction register.

From the Capture state, the TAP transitions to either the Shift or Exit1 state. Normally, the Shift state follows the Capture state so that test data or status information can be shifted out for inspection and new data shifted in. Following the Shift state, the TAP either returns to the Run-Test/Idle state via the Exit1 and Update states or enters the Pause state via Exit1. The reason for entering the Pause state is to temporarily suspend the shifting of data through either the selected data register or instruction register while a required operation, such as refilling a tester memory buffer, is performed. From the Pause state, shifting can resume by re-entering the Shift state via the Exit2 state or be terminated by entering the Run-Test/Idle state via the Exit2 and Update states.

Upon entering the data register scan or instruction register scan blocks, shadow latches in the selected scan path are forced to hold their present state during the capture and shift operations. The data being shifted into the selected scan path is not output through the shadow latch until the TAP enters the Update-DR or Update-IR state. The Update state causes the shadow latches to update (or parallel load) with the new data that has been shifted into the selected scan path.

Figure 3-4 shows the TAP control output signals, along with the instruction register and data register interconnects.

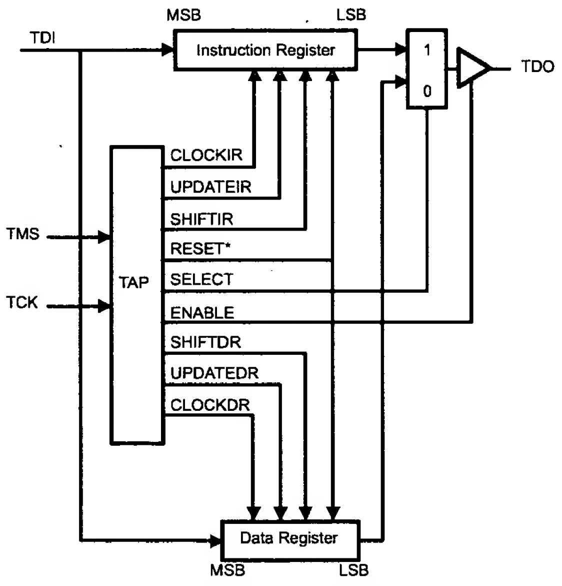


Figure 3-4. TAP Control Output Interconnect Diagram

IEEE Std 1149.1 Registers

This section contains descriptions of the required and optional registers specified in IEEE Std 1149.1-1990.

Instruction Register (Required)

The instruction register is responsible for providing the address and control signals required to access a particular data register in the scan path. The instruction register is accessed when the TAP receives an instruction register scan protocol. During an instruction register scan operation the SELECT output from the TAP (Figure 3-4) selects the output of the instruction register to drive the TDO pin. A general instruction register architecture is shown in Figure 3-5.

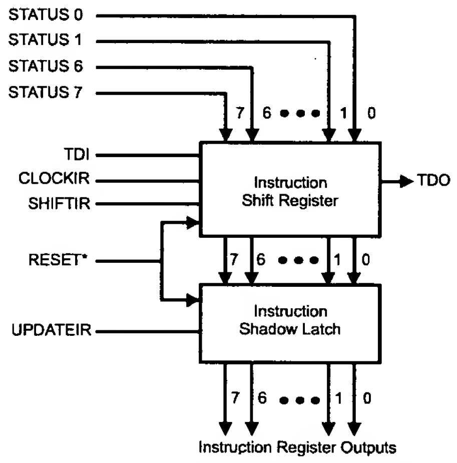


Figure 3-5. General Instruction Register Architecture

The instruction register consists of an instruction shift register and an instruction shadow latch. The instruction shift register (Figure 3-5) consists of a series of shift register bits arranged to form a single scan path between the TDI and TDO pins of the host IC. During instruction register scan operations, the TAP exerts control via the instruction register shift enable (SHIFTIR) and instruction register clock

(CLOCKIR) signals to cause the instruction shift register to preload status information and shift data from TDI to TDO. Both the preload and shift operations occur on the rising edge of TCK; however, the data shifted out from the host IC from TDO occurs on the falling edge of TCK. The status inputs are user-defined observability inputs, except for the two least-significant bits, which are always 01 for scan-path testing purposes. (The instruction register has a minimum length of two bits.)

The instruction shadow register (Figure 3-5) consists of a series of latches, one latch for each instruction shift register bit. During an instruction register scan operation, the latches remain in their present state. At the end of the instruction register scan operation, the instruction register update (UPDATEIR) input updates the latches with the new instruction installed in the instruction shift register. When activated, the RESET* input sets the instruction shadow register to the value of the *BYPASS* instruction (or *IDCODE* instruction, if it is supported). This forces the device into its normal functional mode and selects the bypass register (or device identification register, if one is present).

Data Registers

IEEE Std 1149.1 requires two data registers; boundary-scan register and bypass register, with a third, optional, device identification register. Additional user-defined data registers can be included. The data registers are arranged in parallel from the primary TDI input to the primary TDO output. The instruction register supplies the address that allows one of the data registers to be accessed during a data register scan operation. During a data register scan operation, the addressed scan register receives TAP control via the data register shift enable (SHIFTDR) and data register clock (CLOCKDR) inputs to preload test response and shift data from TDI to TDO. During a data register scan operation, the SELECT output from the TAP (Figure 3-4) selects the output of the data register to drive the TDO pin. When one scan path in the data register is being accessed, all other scan paths remain in their present state.

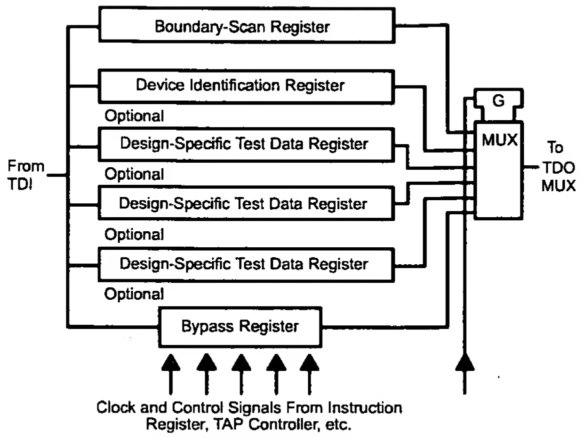

Boundary-Scan Register
Device Identification Register
Optional
Design-Specific Test Data Register
Optional
Design-Specific Test Data Register
Optional
Design-Specific Test Data Register
Optional
Bypass Register
From TDI
G
MUX
To TDO MUX
Clock and Control Signals From Instruction Register, TAP Controller, etc.

Figure 3-6. Test Data Register Architecture

Boundary-Scan Register — The boundary-scan register (BSR) consists of a series of boundary-scan cells (BSCs) arranged to form a scan path around the boundary of the host IC. The BSCs provide the controllability and observability features required to perform boundary-scan testing as described in the *Boundary-Scan Overview* section of this chapter. Shadow latches in the BSCs, driving the NDO outputs remain in their present state during a data register scan operation. At the end of a data register scan operation, the data register update (UPDATEDR) input updates the shadow latches with the new boundary test pattern to be applied from the NDO outputs of the BSCs. Figure 3-7 shows a conceptual view of a control-and-observe BSC.

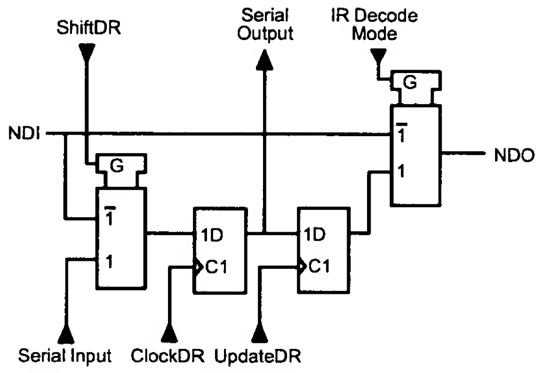


Figure 3-7. Conceptual View of a Control-and-Observe BSC

Bypass Register (Required) — The bypass register consists of a single scan register bit. When selected, the bypass register provides a single-bit scan path between TDI and TDO. Thus, the bypass register allows abbreviating the scan path through devices that are not involved in the test. The bypass register is selected when the instruction register is loaded with a pattern of all ones to satisfy the IEEE Std 1149.1 *BYPASS* instruction requirement.

Device Identification Register (Optional) — The device identification register is an optional register, defined by IEEE Std 1149.1, to identify the device's manufacturer, part number, revision, and other device-specific information. Figure 3-8 shows the bit assignments defined for the device identification register. These bits can be scanned out of the device identification register after its selection.

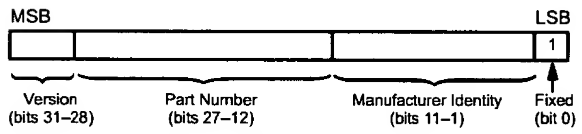


Figure 3-8. Device Identification Register Structure

Although the device identification register is optional, IEEE Std 1149.1 has dedicated an instruction to select this register. The device identification register is selected when the instruction register is loaded with the *IDCODE* instruction, the bit code of which is defined by the vendor.

Manufacturer's identity codes (bit1 through bit11) are assigned, maintained, and updated by the EIA/JEDEC office. Any company can be added to the *JEDEC Standard Manufacturer's Identification Code* (Publication JEP106) by request to the JEDEC office at 202-457-4973.

IEEE Std 1149.1 Required Instructions

IEEE Std 1149.1 defines nine test instructions. Of the nine instructions, three are required and six are optional. The following subsections contain brief descriptions of each required test instruction.

BYPASS Instruction

The required *BYPASS* instruction allows the IC to remain in a functional mode and selects the bypass register to be connected between TDI and TDO. The *BYPASS* instruction allows serial data to be transferred through the IC from TDI to TDO without affecting the operation of the IC. The bit code of this instruction is defined as all ones by IEEE Std 1149.1.

SAMPLE/PRELOAD Instruction

The required *SAMPLE/PRELOAD* instruction allows the IC to remain in its functional mode and selects the boundary-scan register to be connected between TDI and TDO. During this instruction, the boundary-scan register can be accessed via a data scan operation, to take a sample of the functional data entering and leaving the IC. This instruction is also used to preload test data into the boundary-scan register before loading an *EXTEST* instruction. The bit code for this instruction is defined by the vendor.

EXTEST Instruction

The required *EXTEST* instruction places the IC into an external boundary-test mode and selects the boundary-scan register to be connected between TDI and TDO. During this instruction, the boundary-scan register is accessed to drive test data off-chip via the boundary outputs and receive test data off-chip via the boundary inputs. The bit code of this instruction is defined as all zeroes by IEEE Std 1149.1.

IEEE Std 1149.1 Optional Instructions

The following subsections contain brief descriptions of the optional IEEE Std 1149.1 instructions.

INTEST Instruction

The optional *INTEST* instruction places the IC in an internal boundary-test mode and selects the boundary-scan register to be connected between TDI and TDO. During this instruction, the boundary-scan register is accessed to drive test data on-chip via the boundary inputs and receive test data on-chip via the boundary outputs. The bit code of this instruction is defined by the vendor.

RUNBIST Instruction

The optional *RUNBIST* instruction places the IC in a self-test mode, enables a comprehensive self-test of the IC's core logic, and selects a user-specified data register to be connected between TDI and TDO. During this instruction, the boundary outputs are controlled so that they cannot interfere with neighboring ICs during the *RUNBIST* operation. Also, the boundary inputs are controlled so that external signals cannot interfere with the *RUNBIST* operation. The bit code of this instruction is defined by the vendor.

CLAMP Instruction

The optional *CLAMP* instruction sets the outputs of an IC to logic levels determined by the contents of the boundary-scan register and selects the bypass register to be connected between TDI and TDO. Before loading this instruction, the contents of the boundary-scan register can be preset with the *SAMPLE/PRELOAD* instruction. During this instruction, data can be shifted through the bypass register from TDI to

TDO without affecting the condition of the outputs. The bit code of this instruction is defined by the vendor.

HIGHZ Instruction

The optional *HIGHZ* instruction sets all outputs (including two-state as well as three-state types) of an IC to a disabled (high-impedance) state and selects the bypass register to be connected between TDI and TDO. During this instruction, data can be shifted through the bypass register from TDI to TDO without affecting the condition of the IC outputs. The bit code of this instruction is defined by the vendor.

IDCODE Instruction

The optional *IDCODE* instruction allows the IC to remain in its functional mode and selects the optional device identification register to be connected between TDI and TDO. The device identification register (see Figure 3-8) is a 32-bit shift register containing information regarding the IC manufacturer, device type, and version code. Accessing the device identification register does not interfere with the operation of the IC. Also, access to the device identification register should be immediately available, via a TAP data-scan operation, after power-up of the IC or after the TAP has been reset using the optional TRST* pin or by otherwise moving to the Test-Logic-Reset state. The bit code of this instruction is defined by the vendor.

USERCODE Instruction

The optional *USERCODE* instruction allows the IC to remain in its functional mode and selects the device identification register to be connected between TDI and TDO. During the *USERCODE* instuction, the optional 32-bit device identification register captures user-defined information about the IC. Accessing the device identification register does not interfere with the operation of the IC. The bit code of this instruction is defined by the vendor.

Obtaining IEEE Std 1149.1-1990

To learn more about IEEE Std 1149.1, please refer to the publications, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993) and *Supplement to IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Std 1149.1b-1994. These documents are available through IEEE (1-800-678-IEEE). IEEE Std 1149.1-1990/IEEE Std 1149.1a-1993 is also available from Texas Instruments on CD-ROM (see Chapter 9, *Other Support and Learning Products*).

Chapter 4
Using DFT in ASICs

The concern most often voiced by application-specific integrated circuit (ASIC) users is that of testability. This chapter is intended to provide an understanding of ASIC testability that can be used for developing test strategies when designs are being initiated.

Design-for-Test Considerations

Designing testability into any circuit affects the hardware to some degree. Additional logic will probably have to be added. This additional logic increases the amount of silicon required to implement the design. The savings from enhanced testability do not usually show up until the cycle time and testing cost of the circuit and its end system are analyzed.

Fault simulation is an important part of designing for testability. This technique enables you to evaluate your test patterns to determine whether these patterns detect faults. Faults may occur during either the design-tooling stage or the circuit-fabrication stage. A fault simulator uses fault models, such as a node shorted to power (stuck-at-one) or a node shorted to ground (stuck-at-zero), and compares the response of a fault-free circuit with the response of a faulty circuit. If the response of the fault-free circuit is different than the response of the faulty circuit, then the test patterns have detected the fault.

By faulting all the nodes in the circuit, the fault simulator produces the test-pattern fault grade or fault coverage. The fault coverage is the percentage of faults detected among the total faults. The higher the fault coverage, the better the test pattern set separates a faulty circuit from a fault-free circuit. After determining which faults have not been detected by the current set of test patterns, additional test patterns can be generated to detect those faults.

Adoption of design-for-test principles early in the design process ensures the maximum testability for the minimum effort. These guidelines emphasize that test is a part of the design flow, not a process that is done at the end of the design cycle.

Three basic elements must come together to make a successful ASIC circuit:

- Logic design, including schematic capture, library selection, synthesis, and simulation
- Logic testability, including fault-detection and test-application criteria within predefined cost and time-scale budgets
- Vendor's manufacturing capability, including processing and packaging

The Need for Testability

Most engineers involved in the design of ASIC devices are familiar with the trade-offs between gate arrays, standard cells, and full custom devices. They are also familiar with the vendor selection process. The aspect of test capability and testability is often overlooked.

Testability could be ignored when typical designs were only a few thousand gates. These designs were implemented first and then turned over to a test engineer or to a vendor to create a test program for production. As design complexities increased, this approach to testing became futile. Successful high-density ASIC design and manufacturing demand that circuits be built with testability incorporated into the design process.

Although testability imposes additional constraints in the design phase, design verification and test can be unmanageable, if ignored until the design is completed and testability is handled as a postdesign insertion. In fact, the design constraints are overwhelmingly balanced by improved testability, which adds value to the device throughout manufacturing and system life.

Test-Time Cost

Test cost, as it relates to time, is a simple calculation. Most commercial testers cost between $2 million and $3 million. Under normal circumstances, the tester depreciation, plant, operator, and support personnel costs are between 10 and 20 cents per test second.

Brute-force test approaches often generate a large number of test patterns. Since test patterns are run at multiple power-supply values and possibly at multiple temperatures, inefficient pattern sets can severely impact the test costs of a complex ASIC device.

Time to Market

Surveys indicate that 40 percent of the development cycle time for an ASIC device is required for test insertion and test-pattern generation. This figure is expected to increase as device complexity increases. The intent of a design-for-test strategy is to achieve high fault-detection test programs in reduced time (Figure 4-1). The obvious cycle-time reductions result from designed-in testability (elimination of iterative redesigns resulting from poor design practices), and from automatic test-pattern generation (ATPG).

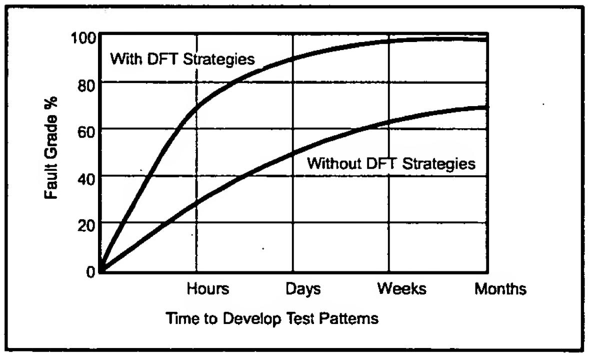


Figure 4-1. Fault Grade Versus Development Time

Figure 4-2 shows the economic relationship between time to market and system manufacturing and field maintenance costs. Point 1 represents the case where market-entry timing forces a constraint on the development time. Since 40 percent of this time is expended in inserting testability, the temptation is to rush to market with devices that are not completely testable or tested. The result is a higher-than-desirable manufacturing and field-maintenance cost. Point 2 represents the case where DFT and ATPG techniques are employed to develop devices that are completely tested. This situation allows an economic optimum that is more favorable to long-term manufacturing and maintenance costs.

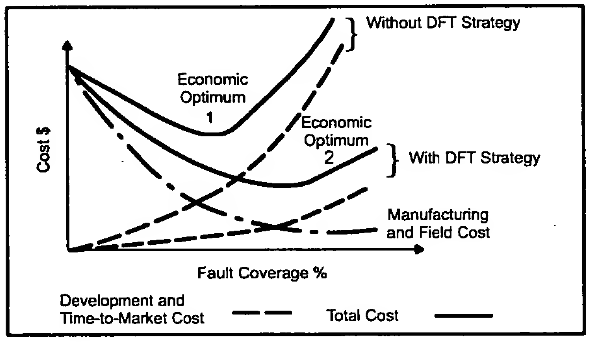


Figure 4-2. Economic Trade-Off for a Testable Design

A less obvious result of a DFT strategy is the reduction of debug time. The designer must make certain assumptions about system requirements. Often, a new device does not work in the system environment and requires debugging. If the device is designed for controllability and observability access, the debugging process is enhanced. Conversely, if these two features are overlooked, debugging and manufacturing can be significantly harder to accomplish, if not impossible. Oscilloscopes and logic analyzers are not very effective in debugging systems utilizing complex ASIC devices in a surface-mount environment.

Fault Coverage and Cost of Ownership

Figure 4-3 shows the trade-off between time to market and manufacturing and field-maintenance costs. The horizontal factor on this figure is fault coverage. The relationship between fault coverage and device defect level is well documented.

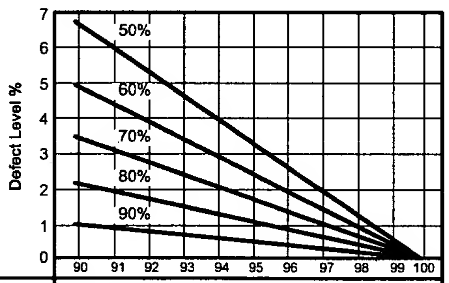
Figure 4-3 is a plot of the relationship modeled by T. W. Williams for fault coverages of 90 percent, or greater.

The Williams model is:

$$D = [1 - Y^{(1-T)}] \times 100$$

Where:

- D = Defect level in percent
- Y = Theoretical functional process yield
- T = Fault coverage of the test program used



	90	91	92	93	94	95	96	97	98	99	100
50% Process Yield —	6.70	6.04	5.39	4.74	4.07	3.41	2.73	2.08	1.38	0.89	.00
60% Process Yield —	4.98	4.48	4.00	3.51	3.02	2.52	2.02	1.52	1.01	0.51	.00
70% Process Yield —	3.50	3.16	2.81	2.47	2.12	1.77	1.42	1.06	0.71	0.36	.00
80% Process Yield —	2.21	1.99	1.77	1.55	1.33	1.11	0.89	0.67	0.45	0.22	.00
90% Process Yield —	1.05	0.94	0.84	0.73	0.63	0.53	0.42	0.32	0.21	0.11	.00

Fault Coverage %

Figure 4-3. Defect Level Versus Fault Coverage

To briefly explore the Williams model, assume that the ASIC vendor has a silicon and assembly process yield that is

70 percent. If the fault grade of the test program is also 70 percent, the defect level is projected to be 10.1 percent, or 101,000 ppm (this is outside the limits of the chart and was calculated). At a fault grade of 90 percent, the defect level is projected to be 3.5 percent or 35,000 ppm.

A study of the model shows that the process yield becomes an insignificant term when the fault coverage of the test program is very close to 100 percent.

Motorola and Delco performed a study in 1980 that supports the Williams model. Their experimental results are shown in Figure 4-4. A fault coverage of 99.9 percent was required to obtain defect levels in the range of 100 ppm.

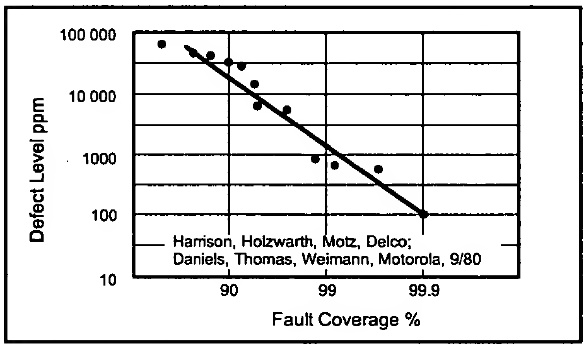


Figure 4-4. Motorola and Delco Study Results

Figure 4-5 shows the maximum allowable ASIC defect rate to achieve a goal PCB defect rate as a function of the number of ASIC devices per board assembly. Note that for multiple-device PCB designs, a goal of 500 ppm requires ASIC defect levels in the range of 100 to 200 ppm.

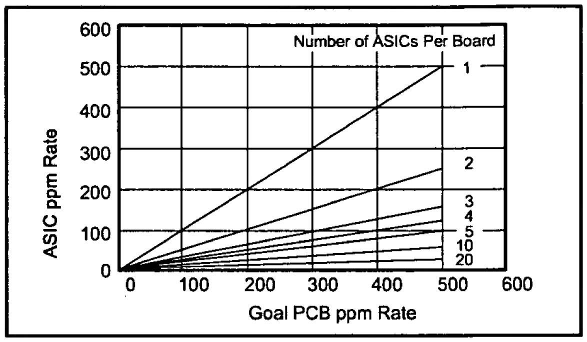


Figure 4-5. ASIC ppm Versus PCB ppm Rate

Theoretical and experimental studies conclude that a high-fault-grade test-pattern set is required for low-defect-level ASIC devices. This type of pattern set is nearly impossible to obtain through brute force. The requirements for a high-fault-grade pattern set are:

- ATPG tool
- Fault grader
- A testable design meeting the constraints of the ATPG tool

As stated earlier, a design-for-test strategy has performance and area costs. Now, the cost of new tools has been added. Benefits such as lower test costs and reduced time to market have been mentioned. These benefits are real, but often hard to quantify. Reduced cost of ownership is another major benefit and is easy to quantify.

Figure 4-6 shows what is commonly referred to as the "cost-of-ownership, order-of-magnitude relationship". It shows that each company has a cost associated with finding a defect in a packaged device before it has entered the assembly process. This cost can be calculated easily. The cost of finding a defective device after assembly onto a PCB is an order-of-magnitude more than before assembly. This continues until the cost to discover a defective device in a system at a customer's site is three orders-of-magnitude higher than that of discovery before assembly on to a PCB.

The lowest cost of ownership is achieved by finding defective units before they are shipped from the vendor.

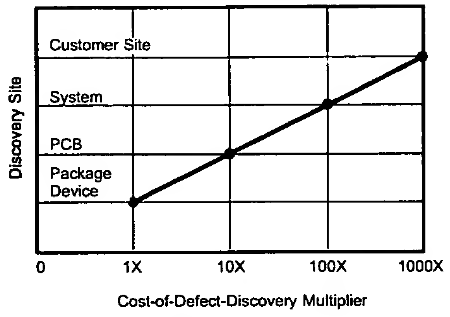


Figure 4-6. Cost of Ownership

The previous discussions lead to the conclusion that the lowest cost of ownership can be obtained by providing the ASIC vendor with an efficient, high-fault-detection set of test vectors. These DFT methodologies provide lower cost of ownership with the added benefit of reduced time to market.

Developing Testability Strategies

The following strategies outline the process of design for test.

1. Select a technology.

   When selecting a technology or vendor, make sure there is enough performance and gate-count margin to allow the insertion of testability.

2. Commit to testability design practices.

   Testability design practices work. Commit to use them, and review them with the design team before the design begins. Add a testability commitment to the design requirements document developed for the design project. Make testability audits part of the design review process.

3. Establish a fault-grade requirement.

   The fault-grade requirement usually can be provided by the manufacturing or quality organization. Establish this requirement before the first design review. Add the fault-grade requirement to the design requirements document. This requirement drives many of the decisions that follow in the development of the test strategy.

Many companies consider the fault-grade requirement to be an index of device cost of ownership. Failure to achieve it impacts profits throughout the lifetime of the device.

4. Decide if IEEE Std 1149.1 is a system requirement.

   When implemented in an ASIC device, IEEE Std 1149.1 allows test of the interconnect between devices on a PCB through a four-pin test bus. If IEEE Std 1149.1 is selected, the four dedicated test pins also can be used to control core-logic test techniques such as built-in self-test (BIST), internal scan, on-chip emulation, and boundary-scan through the test access port (TAP) and instruction register.

5. Select an ASIC testability approach based on gate density.

   - Designs with fewer than 10K gates

     Designs with fewer than 10K gates are not generally complex enough to require structured test approaches. The overhead impact is usually too high to justify them. Nonstructured, good design practices are usually sufficient.

   - Designs with more than 10K gates, but fewer than 20K gates

     Structured techniques should be considered for designs in this density. Nonstructured, good design practices are probably sufficient for highly combinatorial circuits without memory. Structured approaches should be considered as complexity is increased by the addition of sequential circuits, feedback, and memory. Consider boundary-scan testing for reduced cycle times and high fault grades.

   - Designs with more than 20K gates

     The complexity of circuits this dense usually requires structured approaches to achieve high fault grades. At this density, it is often hard to control or observe deeply embedded circuits. The overhead associated with structured testability approaches is acceptable at this density.

6. Choose structured tools.

   Scan testing is typically the preferred structured approach for sequential logic. The available scan choices are:

   - Clocked scan
   - Multiplexed flip-flop scan
   - Level-sensitive scan-design (LSSD)
   - Parallel scan chains
   - Partial scan

7. Establish a diagnostic functional-pattern set to expedite debug.

   This is an important step in decreasing the time to market for an ASIC device. The purpose of this pattern set is to isolate circuitry for analysis.

8. Generate high fault-grade test patterns.

   The fault grade of a test pattern set determines the best possible quality level attainable with that set of patterns.

9. Simulate test patterns and timing.

   Two types of simulation are required during development. Logic simulation verifies both functionality and performance of the device. Test-pattern simulation produces the information needed to verify the test patterns in a tester environment.

Figure 4-7 contains a flow chart that steps through the design-for-test process.

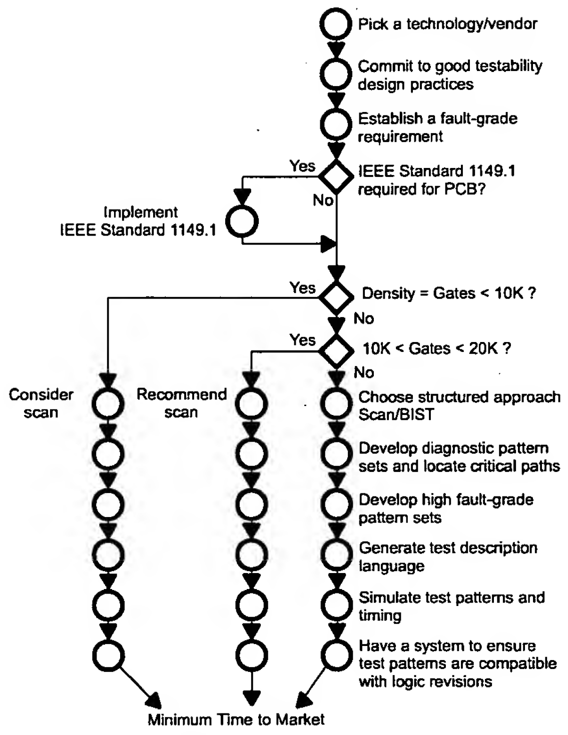

Pick a technology/vendor
Commit to good testability design practices
Establish a fault-grade requirement
IEEE Standard 1149.1 required for PCB?
Yes
No
Implement IEEE Standard 1149.1
Density = Gates < 10K ?
Yes
No
10K < Gates < 20K ?
Yes
No
Consider scan
Recommend scan
Choose structured approach Scan/BIST
Develop diagnostic pattern sets and locate critical paths
Develop high fault-grade pattern sets
Generate test description language
Simulate test patterns and timing
Have a system to ensure test patterns are compatible with logic revisions
Minimum Time to Market

Figure 4-7. Testability Development Flow

Chapter 5
Data Formats

Several data formats have emerged to make IEEE Std 1149.1 successful and well supported by tools. This chapter discusses the most widely accepted data formats that support IEEE Std 1149.1 — BSDL, HSDL, and SVF.

Boundary-Scan Description Language (BSDL)

In 1990, IEEE Std 1149.1 was approved and implementation of the standard accelerated. As more people became aware of and used the standard, the need for a standard method for describing IEEE Std 1149.1-compatible devices was recognized. The IEEE Std 1149.1 working group established a subcommittee to develop a device description language to address this need.

The subcommittee has since developed an industry standard language called Boundary-Scan Description Language (BSDL). BSDL is a subset and standard practice of VHDL (VHSIC Hardware Description Language) that describes how IEEE Std 1149.1 is implemented in a device and how it operates. BSDL captures the essential features of any IEEE Std 1149.1 implementation. BSDL has been formally adopted as part of IEEE Std 1149.1b-1994.

IEEE Std 1149.1 is a structured design-for-test approach well suited for tools and automation. Tools developed to support the standard can control the test access port (TAP). If they know how the boundary-scan architecture was implemented in the device, such tools can also control the I/O of the device. BSDL provides a standard machine- and human-readable data format for describing how IEEE Std 1149.1 boundary-scan architecture is implemented in a device.

How BSDL Is Used

Many IEEE Std 1149.1 tools already on the market support BSDL as a data input format. These tools offer different capabilities to customers implementing IEEE Std 1149.1 into their designs, including board interconnect automatic test-pattern generation (ATPG) and automatic test equipment (ATE).

When tools that support BSDL are used, such BSDL is often received from the semiconductor vendor. This can result in significant time and cost savings.

Teradyne estimates that to create in-circuit test patterns for a leading microprocessor normally can take as long as seven weeks:

- One week to study the device
- Four weeks to develop in-circuit test patterns
- Two weeks to verify the patterns on ATE

The development cost estimate for this approach is $14,000.

If the microprocessor supports IEEE Std 1149.1, and the BSDL is supplied by the vendor, the time to develop in-circuit test patterns is less than two hours (less than $100) using today's tools.

Elements of BSDL

A BSDL description for a device consists of the following elements:

- Entity descriptions
- Generic parameter
- Logical port description
- Use statement(s)
- Component conformance statement
- Pin mapping(s)
- Scan port identification
- Instruction register description
- Optional register decription
- Register access description
- Boundary register description

Entity Descriptions — The entity statement names the entity, such as the device name (e.g., SN74BCT8245A). An entity description begins with an entity statement and terminates with an end statement.

```
entity XYZ is
  {statements to describe the entity go here}
end XYZ;
```

Generic Parameter — A generic parameter is a parameter that can come from outside the entity, or it can be defaulted, such as a package type (e.g., “DW”).

```
generic (PHYSICAL_PIN_MAP : string := "DW");
```

Logical Port Description — The port description gives logical names to the I/O pins (system and TAP pins), and denotes their nature, such as input, output, bidirectional, linkage (analog or power supply/return) and so on.

```
port (OE:in bit;
  Y:out bit_vector(1 to 3);
  A:in bit_vector(1 to 3);
  GND, VCC, NC:linkage bit;
  TDO:out bit;
  TMS, TDI, TCK:in bit);
```

Use Statement(s) — The use statement refers to external definitions found in packages and package bodies.

```
use STD_1149_1_1994.all;
```

Component Conformance Statement — The component conformance statement indicates the latest issue of IEEE Std 1149.1 to which the device conforms.

```
attribute COMPONENT_CONFORMANCE of XYZ : entity is
  "STD_1149_1_1993";
```

Pin Mapping(s) — The pin mapping provides a mapping of logical signals onto the physical pins of a particular device package.

```
attribute PIN_MAP of XYZ : entity is
PHYSICAL_PIN_MAP;
constant DW:PIN_MAP_STRING:=
  "OE:1, Y:(2,3,4), A:(5,6,7), GND:8, VCC:9,  " &
      "TDO:10, TDI:11, TMS:12, TCK:13, NC:14";
```

Scan Port Identification — The scan port identification statements define the device’s TAP.

```
attribute TAP_SCAN_IN of TDI : signal is TRUE;
attribute TAP_SCAN_OUT of TDO : signal is TRUE;
attribute TAP_SCAN_MODE of TMS : signal is TRUE;
attribute TAP_SCAN_CLOCK of TCK : signal is (50.0e6,
    BOTH);
```

Instruction Register Description — The instruction register description identifies the device-dependent characteristics of the instruction register.

```
attribute INSTRUCTION_LENGTH of XYZ : entity is 2;
attribute INSTRUCTION_OPCODE of XYZ : entity is
  "BYPASS (11),"&
  "EXTEST (00),"&
  "SAMPLE (10),"&
  "IDCODE (01)"
attribute INSTRUCTION_CAPTURE of XYZ : entity is
    "01";
```

Optonal Register Description — The optional register description identifies the values captured in the device identification register for the optional *IDCODE* and *USERCODE* instructions, if supported.

```
attribute IDCODE_REGISTER of XYZ : entity is
"01010100000011111100000000101111";
```

Register Access Description — The register access description defines which register is placed between TDI and TDO for each instruction.

```
attribute REGISTER_ACCESS of XYZ : entity is
  "BOUNDARY (EXTEST, SAMPLE),"&
  "BYPASS (BYPASS)";
```

Boundary Register Description — The boundary register description contains a list of boundary-scan cells, along with information regarding the cell type and associated control.

```
attribute BOUNDARY_LENGTH of XYZ : entity is 7;
attribute BOUNDARY_REGISTER of XYZ : entity is
  "0 (BC_1, Y(1), output3, X, 6, 0, Z),"&
  "1 (BC_1, Y(2), output3, X, 6, 0, Z),"&
  "2 (BC_1, Y(3), output3, X, 6, 0, Z),"&
  "3 (BC_1, A(1), input , X),"&
  "4 (BC_1, A(2), input , X),"&
  "5 (BC_1, A(3), input , X),"&
  "6 (BC_1, OE , input , X),"&
  "6 (BC_1, *  , control, 0)";
```

Verifying BSDL Accuracy

Creating a BSDL file that is syntactically and semantically correct is only the beginning. A syntactically and semantically correct BSDL file can still contain descriptive errors and result in time-consuming debugging of board-level tests and diagnostics. A BSDL file must be validated (compared) against the silicon.

Potential BSDL Errors

As with any hand-entered data, typographical errors potentially exist. BSDL contains many commas and semicolons that contribute error possibilities. Fortunately, syntax and semantics errors can easily be identified and corrected using syntax and semantics checking tools during the authoring process.

Other common errors are:

- Wrong pinout
- Wrong cell types
- Wrong boundary-scan register order
- Wrong boundary-scan register length
- Wrong instruction register opcodes
- Wrong control cell locations
- Wrong control cell disable value
- Wrong I/O pin control cell
- Wrong identification code value
- Wrong capture-IR value

Typographical errors or device documentation errors can result in implementation errors.

How to Receive the BSDL Specification

Contact the IEEE (1-800-678-IEEE) and refer to *Supplement to IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Std 1149.1b-1994 to receive a copy of the Boundary-Scan Description Language (BSDL) specification.

Obtaining BSDL for TI Devices

Texas Instruments makes catalog-device BSDL files available to customers through the World-Wide Web. As of this writing, these files can be located on the Texas Instruments IEEE Std 1149.1/JTAG/Boundary-Scan Silicon Products page located at URL http://www.ti.com/sc/docs/jtag/silicon.htm; or, on the main Texas Instruments home page at URL http://www.ti.com. Search for keyword BSDL.

Hierarchical Scan Description Language (HSDL)

Texas Instruments developed the hierarchical scan description language (HSDL) to complement BSDL, using the same subset of VHDL statements as BSDL. HSDL picks up where BSDL stops to describe additional attributes of IEEE Std 1149.1 devices and how IEEE Std 1149.1 devices are connected at the board and system levels.

HSDL uses the BSDL entity and package in new ways. Entities in HSDL are used to describe modules as well as devices. A module is any level of architecture above the device level, including boards, multichip modules, backplanes, subsystems, and systems. In addition, HSDL provides two new packages used to indicate that an entity is an HSDL device or module.

BSDL is ideal for describing how IEEE Std 1149.1 is implemented in a device, but stops there. HSDL provides a method for describing how IEEE Std 1149.1 devices are connected at the board, module, and system levels. HSDL serves two needs not addressed by BSDL:

- Describes the test-bus interconnections of IEEE Std 1149.1 at the board or module level
- Improves ease of use and reduces risk during interactive design debug and verification
- Allows descriptions of boards with dynamic and reconfigurable architectures

Elements of HSDL

HSDL module statements use much of the same syntax as BSDL. New statements have been added to describe the members and scan paths of the module and to simplify interactive use.

- Entity description
- Generic parameter
- Logical port description
- Use statement(s)
- [Optional module description(s)]
- [Optional port description(s)]
- Pin mapping(s)

- Scan port identification
- [Optional members description(s)]
- [Optional bus composition(s)]
- Path description
- [Optional member connections]
- [Optional constraint description(s)]
- [Optional design warning]

Entity Description — The entity statement names the entity, such as the module name (e.g., BOARD). An entity description begins with an entity statement and terminates with an end statement.

```
entity BOARD is
  {statements to describe the entity go here}
end BOARD;
```

Generic Parameter — A generic parameter may come from outside the entity or it may be defaulted, such as a package type (e.g., "UNDEFINED").

```
generic (PHYSICAL_PIN_MAP : string := "UNDEFINED");
```

Logical Port Description — The port description gives logical names to the I/O pins (system and TAP pins), and denotes their nature such as input, output, bidirectional, linkage (analog or power supply/return) and so on.

```
port (TDI:in bit;
  TDO:out bit;
  TMS:in bit;
  TCK:in bit;
  GND:linkage bit);
```

Use Statement(s) — The use statement refers to external definitions found in packages and package bodies.

```
use STD_1149_1_1994.all;
use HSDL_module.all;
```

Pin Mapping(s) — The pin mapping provides a mapping of logical signals onto the physical pins of a particular entity.

```
attribute PIN_MAP of BOARD : entity is
PHYSICAL_PIN_MAP;
  constant PINOUT1 : PIN_MAP_STRING :=
  "TDI:1, TDO:2, TMS:3, TCK:4, GND:5";
```

Scan Port Identification — The scan port identification statements define the entity's TAP.

```
attribute TAP_SCAN_IN of TDI : signal is TRUE;
attribute TAP_SCAN_OUT of TDO : signal is TRUE;
attribute TAP_SCAN_MODE of TMS : signal is TRUE;
attribute TAP_SCAN_CLOCK of TCK : signal is (5.0e6,
            LOW);
```

Members Description (Optional) — Members represent devices or other modules that are on the module. Usually members represent components, but some boards may contain scannable daughtercards, card slots, etc. that require member modules to describe them.

```
attribute MEMBERS of BOARD : entity is
  "U1 (XYZ1, DW),"&
  "U2 (XYZ2, DW)";
```

Bus Composition (Optional) — Buses in an HSDL module can be built of module buses, member module buses, member device buses, and member device test registers.

```
attribute BUS_COMPOSITION of BOARD : entity is
  "bus1[4]  (U1.Boundary[3,0]),"&
  "bus2[4]  (U2.Boundary[3,0])";
```

Path Description — Module paths are intended to describe the netlist of TAP signals (scan paths) on the board.

```
constant boardpath1 : STATIC_PATH :=
  "U1, U2";
end BOARD;
```

How to Receive the HSDL Specification

Originally developed by Texas Instruments primarily in support of its ASSET business, HSDL is now supported by ASSET InterTech, which acquired ASSET™ in 1995.

Serial Vector Format (SVF)

Serial Vector Format, commonly referred to as SVF, was jointly developed by Texas Instruments and Teradyne in 1991. SVF is a standard ASCII format for expressing test patterns that represent the stimulus, expected response, and mask data for IEEE Std 1149.1-based tests. The need for SVF arose from the desire to have vendor-independent IEEE Std 1149.1 test patterns that are transportable across a wide selection of simulation software and test equipment — from design verification through field diagnostics.

Boundary-scan test execution is controlled by the sequencing of TAP signals on the pins of the devices. Each device's behavior is determined solely by the states of its TAP pins. Boundary-scan tools must maintain knowledge of the sequences required to exert certain behaviors within a device and where that device is located in the serial scan path.

SVF controls the IEEE Std 1149.1 test bus using commands that transition the TAP from one steady state to another. Rather than describe the explicit state of the IEEE Std 1149.1 bus on every TCK cycle, SVF describes it in terms of transactions conducted between stable states. For instance, the process of scanning in an instruction is described merely in terms of the data involved and the desired stable state to enter after the scan has been completed. The Capture, Update, Pause, etc. states are inferred rather than explicitly represented. The data to be scanned in, expected data out, and compare mask are all grouped in an easily understandable manner. A command is provided to support deterministic navigation of TAP states where required.

In addition to supporting higher level depictions of scan operations, SVF also supports combined serial and parallel operations. This allows SVF to accommodate ATE environments where some stimulus/response is handled via parallel I/O, and serial signals are accessed via an IEEE Std 1149.1-control environment.

SVF also supports the concept of scan offsets. Offsets allow a test to be applied to a component or cluster of logic embedded in the middle of a scan path. In Figure 5-1, a device exists in multiple instances on a board. Serially applied tests were generated by the designer that are available in SVF format. To reuse this test, it is necessary to put all other devices on the scan path into the *BYPASS* mode. The IEEE Std 1149.1 test controller must therefore comprehend 24 instruction register bits (8-bit IR x 3 devices) before and 16 instruction register bits (8-bit IR x 2 devices) after the target device. Once in *BYPASS*, the devices introduce three data register bits before and two data register bits after the target device.

SVF allows a header and trailer to be defined once, which maintains the instruction register and data registers of the nontargeted devices in the desired *BYPASS* state. No modifications are required to the SVF for the device. In Figure 5-1, if the same test were targeted toward another device downstream, this would be accommodated merely by changing the headers and trailers.

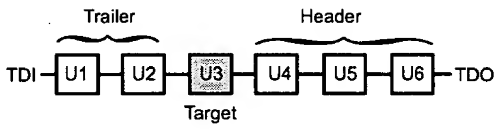


Figure 5-1. Scan Path of Six ICs

The offset approach is capable of installing any instruction register and data register stimulus, provided these values are constant for the entire process of applying the SVF device sequence.

SVF Structure

The SVF file is defined as an ASCII file that consists of a set of SVF statements. Statements are terminated by a semicolon (;) and may continue for more than one line. The maximum number of ASCII characters per line is 256. SVF is not case sensitive, and comments can be inserted into an SVF file after an exclamation point (!) or a pair of slashes (//).

Each statement consists of a command and parameters associated with that specific command. Commands can be grouped into three types: state commands, offset commands, and parallel commands.

State Commands

State commands are used to specify how the test sequences will traverse the IEEE Std 1149.1 TAP state machine. The following state commands are supported:

SDR	Scan data register
SIR	Scan instruction register
ENDDR	Define end state of DR scan
ENDIR	Define end state of IR scan
RUNTEST	Enter Run-Test/Idle state
STATE	Go to specified stable state
TRST	Drive the TRST line to the designated level

SDR performs an IEEE Std 1149.1 data register scan. *SIR* performs an IEEE Std 1149.1 instruction register scan. *ENDDR* and *ENDIR* establish a default state for the bus following any data register scan or instruction register scan, respectively. *RUNTEST* goes to Run-Test/Idle state for a specific number of TCKs. For each of the above commands, a default path through the state machine is used. Each of these commands also terminates in a stable, nonscannable state.

STATE places the bus in a designated IEEE Std 1149.1 stable state. *TRST* activates or deactivates the optional test-reset signal of the IEEE Std 1149.1 bus.

Offset Commands

Offset commands allow a series of SVF commands to be targeted toward a contiguous series of points in the scan path. Examples would be a sequence for executing self-test on a device, or a cluster test where all devices involved in the cluster test are grouped together. The following offset commands are supported:

HDR	Header data for data bits
HIR	Header data for instruction bits
TDR	Trailer data for data bits
TIR	Trailer data for instruction bits

HDR specifies a particular pattern of data bits to be padded onto the front of every data scan. *HIR* specifies the same for the front of every instruction register scan. *TDR* and *TIR* specify data to be injected on the back of each scan. These patterns need only be specified once and are included on each scan unless changed by a subsequent *HDR*, *HIR*, *TDR*, or *TIR* command.

Parallel Commands

Parallel commands are used to map and apply the following commands:

PIO — Specifies a parallel test pattern

PIOMAP — Designates the mapping of bits in the PIO command to logical pin names

Parallel commands allow SVF to combine serial and parallel sequences. *PIOMAP* commands are used by parallel I/O controllers to map data bits in the command into parallel I/O channels using the ASCII logical pin name as a reference. The *PIO* command specifies the execution of a parallel pattern application/sample. SVF does not specify any other properties of parallel I/O such as drive, levels, or skew.

Default State Transitions

SVF uses names for the TAP states that are similar to the IEEE Std 1149.1 TAP state names. Table 5-1 lists the SVF equivalent names for the TAP states.

Table 5-1. SVF TAP State Names

IEEE Std 1149.1 TAP State Name	SVF TAP State Name
Test-Logic-Reset	RESET
Run-Test/Idle	IDLE
Select-DR-Scan	DRSELECT
Capture-DR	DRCAPTURE
Shift-DR	DRSHIFT
Pause-DR	DRPAUSE
Exit1-DR	DREXIT1
Exit2-DR	DREXIT2
Update-DR	DRUPDATE
Select-IR-Scan	IRSELECT
Capture-IR	IRCAPTURE
Shift-IR	IRSHIFT
Pause-IR	IRPAUSE
Exit1-IR	IREXIT1
Exit2-IR	IREXIT2
Update-IR	IRUPDATE

Table 5-2 identifies sample default paths taken when transitioning from one state to a specified new state. For example, if the current state is RESET and DRPAUSE is selected as the end state, the TAP moves from RESET through IDLE, DRSELECT, DRCAPTURE, DREXIT1 to DRPAUSE. One must only specify the current and end states and not each intermediate step. See the TAP controller state diagram on page 3-5 of Chapter 3, *Boundary-Scan Architecture and IEEE Std 1149.1* to determine state transition paths not shown in Table 5-2.

Table 5-2. Stable-State Path Examples

Current State	End State	State Path
RESET	RESET	RESET
RESET	IDLE	RESET
		IDLE
RESET	DRPAUSE	RESET
		IDLE
		DRSELECT
		DRCAPTURE
		DREXIT1
		DRPAUSE
RESET	IRPAUSE	RESET
		IDLE
		DRSELECT
		IRSELECT
		IRCAPTURE
		IREXIT1
		IRPAUSE

SVF Example

The following is an example SVF file:

```
!  Begin Test Program
!  Disable Test Reset line
     TRST OFF;
!  Initialize UUT
     STATE RESET;
!  End IR scans in DRPAUSE
     ENDIR DRPAUSE;
!  End DR scans in DRPAUSE
     ENDDR DRPAUSE;
!  24 bit IR header
     HIR 24 TDI (FFFFFF);
!  3 bit DR header
     HDR 3 TDI (7);
!  16 bit IR trailer
     TIR 16 TDI (FFFF);
!  2 bit DR trailer
     TDR 2 TDI (3);
!  8 bit IR scan, load BIST opcode
     SIR 8 TDI (41) TDO (81) MASK (FF);
! 16 bit DR scan, load BIST seed
     SDR 16 TDI (ABCD);
!  RUNBIST for 95 TCK Clocks
     RUNTEST 95 TCK ENDSTATE IRPAUSE;
! 16 bit DR scan, check BIST status
     SDR 16 TDI (0000) TDO(1234) MASK(FFFF);
!  Enter Test-Logic-Reset
     STATE RESET;
!  End Test Program
```

The test begins by deasserting TRST. The DRPAUSE state is established as the default end state for instruction register scans and data register scans. Twenty-four bits of header and 16 bits of trailer data are specified for instruction register scans. No status bits are checked. Three bits of header data and two bits of trailer data are specified for data register scans.

In the example above, a single device in the middle of the scan is targeted. Notice from the 24-bit IR header (3 x 8-bit IR) and the 3-bit DR header (3 x 1-bit DR) that the targeted device has three devices before it in the scan path. From the 16-bit IR trailer (2 x 8-bit IR) and the 2-bit DR trailer (2 x 2-bit DR), the targeted device has two devices following it in the scan path. After the header and trailer offsets are established, all subsequent scans are the concatenation of the header, scan data, and trailer bits. The targeted device supports BIST, which is initialized by scanning hex 41 into the instruction register, followed by hex ABCD into the selected data register. The BIST in the targeted device is then executed by entering the Run-Test/Idle state for 95 clock cycles. Next, the BIST result is scanned out and the status bits compared against a deterministic value to determine pass/fail.

How to Receive the SVF Specification

Orignally developed by Texas Instruments (with Teradyne) primarly in support of its ASSET business, SVF is now supported by ASSET InterTech, which acquired ASSET™ in 1995.

Chapter 6
Suggested Design-for-Test Flow

The designer of any new product must plan for testing at any time in the life cycle of the product. This process is called design for test (DFT). The test methodology, defined by IEEE Std 1149.1, is used to ease problems associated with both product development and all levels of testing. Once boundary-scan architecture is built into a device, tests developed for that device can be reused after it is in a product. Figure 6-1 summarizes the concerns applicable to any new product design.

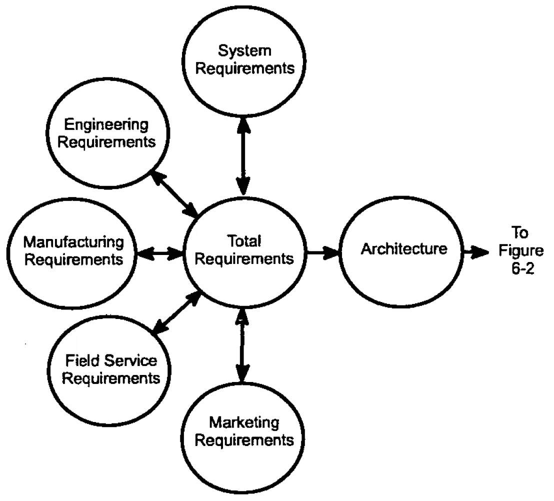


Figure 6-1. Initial DFT Concerns

Test Requirements

Before implementing a boundary-scan test scheme, the designer must define test requirements for the following phases of the product life cycle:

- Design verification
- Manufacturing test
- Field test (also can be used for incoming inspection)

These studies must ensure that the necessary controls are designed into the hardware to support the following requirements:

- During design verification, an engineer requires controllability of the circuitry, both internal to the IC and at board and system levels. Design engineers need controllability and observability of every logic node in the system. If this proves to be unrealistic, reasonable coverage is achievable with partial scan in a boundary-scan design.
- Manufacturing test may include interconnect testing and functional logic validation.
- Field or embedded testing typically may only include the ability to report faults and to isolate a fault to a field-repairable unit (board or subsystem).

For both manufacturing test and field test, IEEE Std 1149.1 logic and test vectors can be used to test clusters or blocks of logic.

Internal IC testing at the board or system level requires a subset of the complete IC logic-verification patterns to validate basic IC logic functions. Each level of hierarchical testing has a different set of fault characteristics. IEEE Std 1149.1 logic can be used throughout the testing life cycle of the product.

Built-In Self-Test (BIST) Methodology

BIST usually consists of special circuitry built as part of an IC's internal design. BIST tests typically perform these functions:

- Pseudorandom pattern generation (PRPG)
- Parallel signature analysis (PSA)
- Serial signature analysis (SSA)
- State machine-based generation/comparison

IEEE Std 1149.1 commands can initiate BIST tests and return the test results.

Internal Scan Test Methodology

Many high-density microprocessors and ASICs now include internal IEEE Std 1149.1-compliant scan circuitry. High-pin-count and fine-pitch packages limit physical access to package pins and devices with high gate count limit observation of the internal logic states. Internal scan, either partial or full, can do the following:

- Provide access to internal registers
- Partition the design
- Reduce the design to combinatorial networks surrounded by scannable registers
- Provide fast, efficient testing of internal logic

Internal scan also allows on-chip emulation control and circuit observability within a device to provide design verification. The emulation capability also can aid in software development to control highly complex functions designed into a device.

Design Effort

Figure 6-2 summarizes the various steps involved in ASIC and board design.

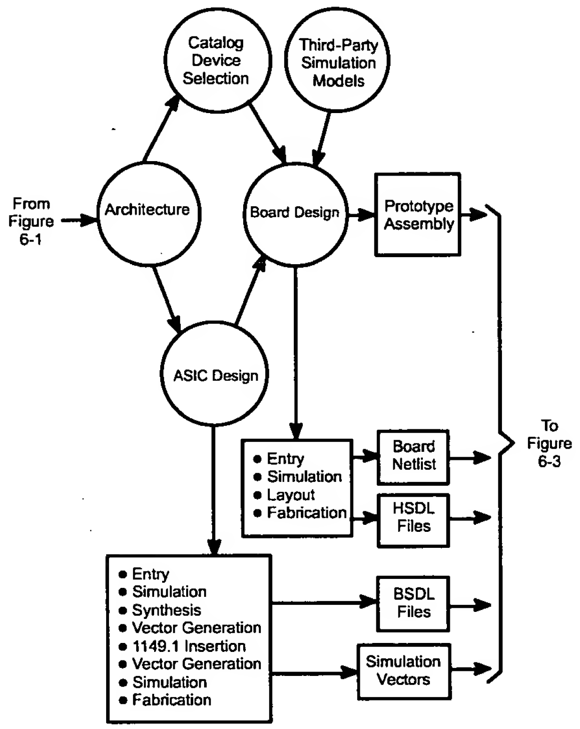


Figure 6-2. Designing Testability for ASICs and Boards

IC Design Implementation

Synthesis of ASICs has become a popular approach to ASIC design. ASIC designers faced with DFT for the first time are concerned about lack of knowledge and the additional design time involved. Today, there are several vendors supporting synthesis and/or insertion tools that can automatically generate the following elements of an IEEE Std 1149.1 architecture:

- Test access port (TAP) controller
- Instruction register
- Instruction decode logic
- Bypass register
- Boundary-scan register

Such tools also can automatically generate test vectors for IEEE Std 1149.1-compliant designs. This means that a designer can quickly and correctly implement basic IEEE Std 1149.1 test circuitry in an ASIC.

If synthesis is not available, most ASIC vendors offer a library of macros that the designer can use to implement IEEE Std 1149.1 into their designs. Using vendor macros requires considerably more effort and knowledge to implement IEEE Std 1149.1 into an ASIC, compared to using synthesis. However, in the absence of synthesis or boundary-scan insertion tools, using vendor macros is significantly easier than implementing boundary-scan architecture using a library that has no such macros. Texas Instruments offers a family of IEEE Std 1149.1-compliant ASIC macros for gate array and standard cell designs.

The ASIC designer is required to provide a Boundary-Scan Description Language (BSDL) file that describes the finished ASIC. This information becomes the basis for tools that automatically create board interconnect test vectors. These vectors can be used by the board designer for debug and validation and by manufacturing test departments for quality assurance.

Several DFT tool vendors plan to implement automatic BSDL creation that is available once the ASIC design has been completed. The creation of a BSDL file by the ASIC designer is straightforward if the design tool used does not automatically create it.

IC Simulation

During IC simulation, the designer can use IEEE Std 1149.1 logic to simplify the simulation patterns required to validate the design. First, patterns must be developed to verify that the IEEE Std 1149.1 interface is operating properly. Simple IR and DR scan patterns can be developed to perform this check. If internal scan is implemented, then simulation pattern sets can be written to test internal logic clusters using the internal scan registers. This partitioned approach simplifies design validation.

Using SVF for IC Design Validation

Many IEEE Std 1149.1-compliant tools use Serial Vector Format (SVF) as the primary test pattern data format and interfaces between simulators and ATE testers also support SVF.

A collection of manually-generated SVF patterns, simulation patterns translated into SVF and ATPG-generated SVF patterns can be used to validate the IEEE Std 1149.1 logic. Initial tests must first verify that serial data can be scanned into and out of the IC. Proper use of internal scan registers to partition the internal functional logic can simplify fault detection and isolation.

IC testing usually is broken into three parts; static functional tests at 1 MHz, at-speed functional tests, and parametric tests. Because IEEE Std 1149.1 architecture relies on serially scanning control and test data, only static functional testing and DC parametric testing can be performed via the IEEE Std 1149.1 architecture. However, the IEEE Std 1149.1 TAP can be used to access user-defined test instructions (such as *RUNBIST*), which can be operated at speed.

INTEST patterns generated by a simulator can be used to validate the internal logic of the IC. By comparing these patterns with simulation results and by observing internal scan register contents, the designer can isolate faulty internal logic. Figure 6-3 summarizes the design validation flow for all products, from IC through system-level products.

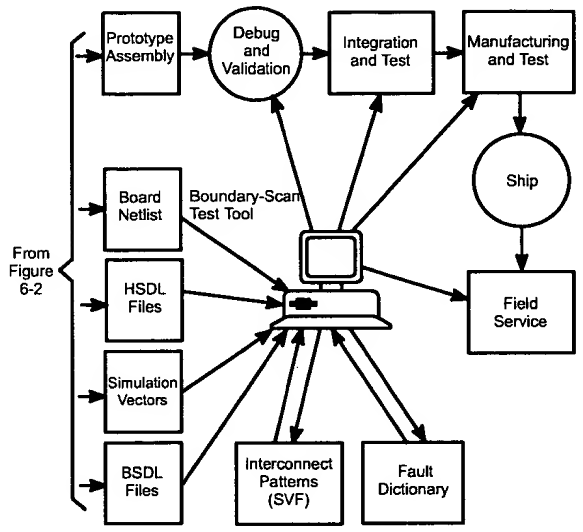


Figure 6-3. Debug and Verification of a Boundary-Scan Design

Data Passed to Board Designer

The board designer needs test data to use during the design debug and validation phase. This data includes:

- BSDL for the ASICs as developed by the ASIC designer
- BSDL for catalog devices from the device vendor

The BSDL that the designer creates for the ASIC must be validated to ensure that it is correct since it will be used by tools that develop more tests for board debug and validation. The new tests are then applicable for manufacturing test and field service test vector generation. BSDL created by a synthesis tool should accurately reflect a device's construction.

Along with the BSDL file for the device, the ASIC designer may also be asked to provide a subset of the ASIC simulation patterns if the ASIC supports the optional *INTEST* instruction. A subset of the simulation patterns may consist of the static (1-MHz) functional patterns. Only a subset of patterns may be needed because it may be impractical to reapply the entire static functional test pattern set at the board level. Many companies have found that they can achieve an acceptable level of confidence that the ASIC is functioning correctly using only a portion of the entire static functional pattern set.

Board Design

Most of the concepts described for IC design also apply to board and system integration. BSDL is used to describe the IEEE Std 1149.1 scan architecture of an IC, and Hierarchical System Description Language (HSDL) is used to describe subsystems and systems. HSDL is used to describe the scan architecture, group discrete signals into logical groups (e.g., ADDR0–15, DATA0–7) and to define constraints.

Constraints

A constraint defines a logic condition that can apply potentially harmful signals to a circuit. IEEE Std 1149.1 gives direct control of all scannable signals, so it is possible to create logic states that functionally would not be entered. For example, bus-arbitration logic can be overridden and dangerous hardware conditions or conflict can occur. An example of a constraint is to inhibit any test pattern that would cause two scannable output nodes to drive a common bus net at the same time.

Partitioned Scan Path

At the board level, the designer must consider how to partition the IEEE Std 1149.1 scan path. Commercial devices are available (such as TI's SN74ACT8997† SPL) to switch a single primary scan path to one or more multiple secondary scan paths. The designer should consider the benefits and penalties associated with partitioned scan paths.

† *SN54ACT8997, SN74ACT8997 Scan Path Linkers With 4-Bit Identification Buses*, Literature Number SCAS157C, Texas Instruments 1996.

Penalties associated with partitioned scan paths include:

- Scan overhead required to switch from one partition to another
- Timing delays in routing TMS (affecting scan frequency)
- Added complexity of the IEEE Std 1149.1 test program

Benefits associated with partitioned scan paths include:

- Shorter scan paths
- Additional fault tolerance
- Somewhat simplified test software

Board Validation/Manufacturing Test

The IEEE Std 1149.1 architecture has the ability to detect, and in many cases, isolate a scan path fault. The instruction register definition requires that the two (2) least-significant bits (those closest to TDO) of a device be, respectively, a logic 1, followed by a logic 0 during an IR scan. This rule allows the IEEE Std 1149.1-compliant bus controller to perform a simple IR scan to prove all devices on the bus can drive a logic 1 and logic 0 on the TDI–TDO path. There are ATPG tools available that automatically generate SVF bus validation patterns that can be applied by the bus controller.

The primary objective of IEEE Std 1149.1 is to allow control and observation of device input and output pins for the purpose of interconnect testing. The reduced size of IC packages has made interconnect testing of PWBs difficult and expensive when using conventional interconnect testing. Commercial tools are available that automatically generate SVF patterns for interconnect testing. Such tools input a board netlist and the BSDL of the scannable components and output an SVF pattern set. This pattern set, along with a fault dictionary lookup table, allows an IEEE Std 1149.1 bus controller to apply the interconnect patterns to the board and detect/isolate faulty interconnections.

After proving correct operation of the IEEE Std 1149.1 bus and board interconnect, the designer can control and observe all boundary-scan nodes. Commercially available boundary-scan test tools allow interactive, macro, or high-level-language programming control for the designer. The board designer also can import and reapply the SVF patterns developed by the IC designer. This reuse of pattern sets provides these benefits:

- Knowledge is passed from one designer to another.
- Overall product development cycle is reduced.
- Using common SVF patterns reduces ambiguity between hierarchical test levels.

The SVF pattern set developed by the IC designer tests an individual IC. The board designer has to be able to relocate the relative scan position of the IC SVF test to match its scan position on the board dynamically. SVF headers and trailers can be manipulated by the boundary-scan test tool to allow such dynamic test relocation.

Summary

As designers become better acquainted with IEEE Std 1149.1, they will no doubt find benefits well beyond those discussed in this chapter. The standard became available just in time to offer solutions and benefits to customers experiencing complex problems introduced by new technology.

The benefits that customers can realize from IEEE Std 1149.1 architecture include:

- Reuse test data throughout the product life cycle
- Apply off-the-shelf components, tools, and ATE solutions from multiple vendors
- Maintain common data base formats for these benefits:
  - Shared common databases between IC, board, and system designers (BSDL, HSDL)
  - Reusable pattern sets (SVF)
  - Reusable test pattern set (TPS) code

Chapter 7
Applications

This chapter presents a number of testing problems and shows how boundary-scan testing and TI products can be used to solve them.

Board-Etch and Solder-Joint Testing

The current approach to detecting board-etch and solder-joint faults in today's electronics industry uses two forms of automatic test equipment (ATE): bare-board testers and digital in-circuit testers. Bare-board test is done after board fabrication. Tests for open-etch circuits between nodes force a voltage onto one circuit node and sense the voltage at another circuit node. This technique is used only on a raw, unassembled board.

An in-circuit test is done after board assembly and checks for:

- Correct orientation of parts
- Correct value and type of parts
- Correct solder joints between all parts and the board

Software models of each part on a board are used to generate the test patterns used by the in-circuit tester. The main assumption is that both board and in-circuit testers require physical access to nodes or points on the board to perform the necessary testing. Both testers require the ability to inject a voltage or current onto a nodal network and sense the voltage or current at another point on the network.

Higher-density packaging technologies, finer-pitch board etch, and more complex digital integrated circuits have made bare-board and in-circuit testing much more expensive.

Solution

Several testability products from Texas Instruments lend themselves to support IC-to-IC continuity testing. The techniques outlined allow detection and, in most cases, isolation of board-etch and solder-joint faults, without forcing any additional requirements on the electrical design engineer. ASIC devices and bus-interface components with boundary-scan cells (BSCs) allow controlled insertion of digital logic values onto a nodal network and provide the means to sample the digital logic values on the nodal network. Broken-etch and solder-joint faults can be detected and isolated in this manner.

Detailed Description

In the simple circuit shown in Figure 7-1, one driver (IC #1) on a network drives digital logic values while the other receivers (ICs #2 and 3) sense changing values on the net. If multiple driver ICs exist on a network, each driver is allowed to drive, in turn, while all receivers sense to detect and isolate a fault to a specific driver or portion of the net.

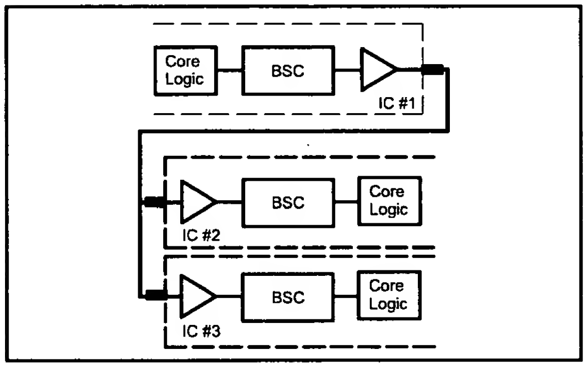


Figure 7-1.   Etch/Interconnect Testing

If an etch is broken on a network or an IC is not soldered to the board properly, then nodes of some ICs are not connected to nodes on others, as shown in Figure 7-2 and Figure 7-3 . When driver IC #1 drives, only IC #2 senses the correct logic value; IC #3 does not.

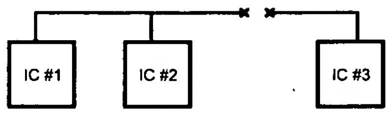


Figure 7-2. ***Open-Etch Condition***

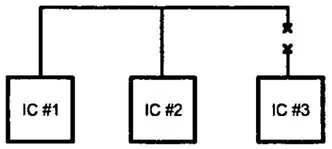


Figure 7-3. ***Open-Solder-Joint Condition***

If an etch is shorted to ground or power due to a solder fault, then none of the ICs are able to drive or sense the correct logic values. See Figure 7-4.

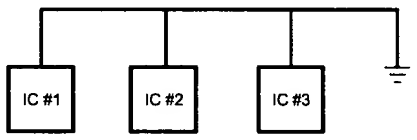


Figure 7-4. ***Short-to-Ground Condition***

The overall coverage of etch- and solder-fault detection and isolation depends on the percentage of a board's circuitry that has boundary-scan architecture. Only networks driven by BSCs or IEEE Std 1149.1-compliant ICs can be tested in this way; nonscan-controlled networks require conventional approaches.

Because the driver and receiver BSCs are in silicon within the ASIC, this technique can be used to test silicon bonding to package pins in the integrated-circuit foundry. A simple bond-out tester can drive a logic value to an input pin of an IC and the sensing function can occur on the BSC buffering the input to the core logic of the ASIC. The reverse procedure can be used on an output pin of an IC; the BSC driving the output of the core logic forces a logic value that the bond-out tester senses and records. See Figure 7-5.

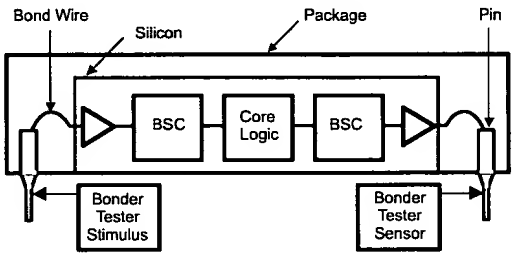


Figure 7-5. Bond-Wire Testing

Summary

The current technique used to detect and isolate board-etch and solder-joint faults remains unaltered. With TI's suite of testability products, board testing can be performed without external digital in-circuit testers. Boundary-scan architecture eliminates the need for physical probe points on printed-circuit boards that contain advanced-packaged, surface-mount integrated circuits. The need to develop part models for in-circuit test generation also is eliminated.

Cluster Testing

Complex digital designs use large-scale integrated circuits and ASICs to accomplish many functions on compact circuit boards. Not all logic functions are integrated, so discrete devices are still used for some functions. Discrete logic seldom includes boundary-scan test architecture due to the expense and complexity; these parts must be tested another way.

Solution

Testing discrete logic functions on a board has to be done from the boundary of the logic cluster. Combinatorial logic is normally driven by and into sequential elements such as a register or a clocked ASIC. By replacing the register with a IEEE Std 1149.1-compliant device, or by using Texas Instruments IEEE Std 1149.1-compliant macro cells at the boundary of the ASIC, the cluster logic may be easily tested. By using members of TI's family of testability products, a higher percentage of circuitry may be tested while reducing the vector count necessary to perform a functional test.

Detailed Description

Figure 7-6 shows a simple circuit incorporating combinational logic sandwiched between sequential logic.

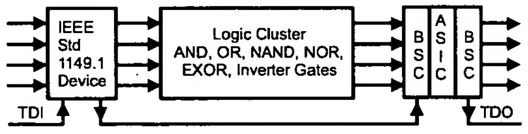


Figure 7-6. Cluster Testing

During normal circuit operation, the IEEE Std 1149.1-compliant device drives the glue logic through to the BSCs in the ASIC. Two methods may be used to perform a test of the logic cluster using the TI family of testability products. There are benefits and drawbacks to both methods.

1. Method one is to scan a fixed-order set of test patterns called *deterministic test vectors* into the IEEE Std 1149.1-compliant device and sample the resultant data at the ASIC boundary.
2. Method two is to command the IEEE Std 1149.1-compliant device to output a pseudorandom pattern of data while the ASIC performs a parallel signature analysis of its inputs.

Deterministic Test Vectors

Deterministic test vectors scanned into the UUT can pinpoint a problem more easily than signature analysis. Since the exact input stimulus is known, the failing response can be detected. Deterministic test vectors perform a complete and meaningful test, but consume a lot of time and enough memory to store all vector data for both stimulus and response.

Signature Analysis

Signature analysis can be performed quickly and virtually automatically. After valid pattern generation and signature analysis seed values are obtained, the resultant signature can be checked against the correct value. This test can be performed at the scan clock speed and requires very little memory for data storage. However, the fault cannot be located without using an extensive fault dictionary.

Signature analysis requires that the IEEE Std 1149.1-compliant device be able to generate stimulus patterns and the ASIC be able to capture a signature of its inputs. This means that appropriate macro cells must be used as outputs of the driving device and as inputs to the boundary of the ASIC. These cells require additional space and additional circuitry to be incorporated into each BSC.

Summary

Using the IEEE Std 1149.1-compliant family of testability products from Texas Instruments enables designers to add features previously unachievable in their designs. It is beneficial to test glue-logic clusters using existing boundary-scan components that surround the logic cluster. This provides added value to the boundary-scan overhead built into a design's larger devices.

Board-Edge Connector Testing

Board-to-board fault detection and isolation within a digital system have historically depended primarily on built-in test (BIT) software. This approach is adequate for detecting functional faults on boards, assuming that the processor executing BIT software can access the boards installed in the system. Backplane faults typically manifest themselves in these ways:

- Broken or bent pins on connectors
- Broken etch on a backplane motherboard
- Termination-network fault
- Bus contention on a bused network

Many bus faults in today's digital processor environments are so catastrophic in nature that the processor executing BIT cannot access the boards installed within the system to isolate the fault.

Solution

In much the same way that boundary scan allows detection and isolation of etch and solder-joint faults on a PWB, surrounding the bus interface of a board with IEEE Std 1149.1-compliant devices allows detection and isolation of bus and connector faults. By placing IEEE Std 1149.1-compliant devices in a controlled test mode, each board, in turn, can drive the backplane bus while the other boards receive the driven logic values.

Detailed Description

Broken etch on a motherboard backplane presents problems similar to that of broken etch on a PWB connecting integrated circuits; some boards cannot communicate to others past the etch break (see Figure 7-7). By placing the testability devices on each board into a controlled test mode, one board at a time can drive logic data onto the bus while the others are receiving logic data. This method provides enough conclusive data to detect and isolate the continuity break.

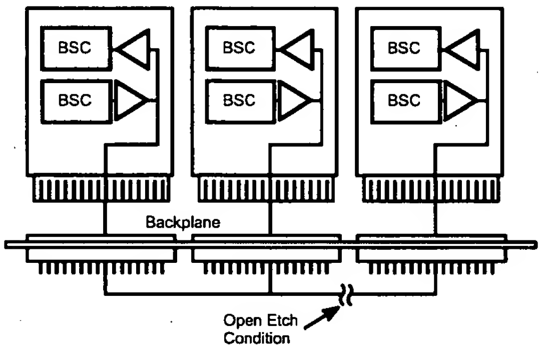


Figure 7-7. Backplane Open-Circuit Fault

A broken connector pin fault can be detected and isolated in the same manner as etch breaks; the board with a bad pin is not able to drive and receive data onto a particular channel on the backplane. See Figure 7-8.

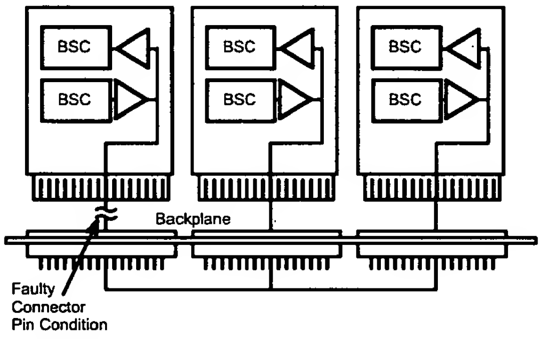


Figure 7-8. PWB Connector Fault

Termination network open or short to V_{CC}/ground shows up as a channel on the backplane that cannot be driven to a certain logic state. The characteristic of this fault could be confused with a bus contention fault. See Figure 7-9.

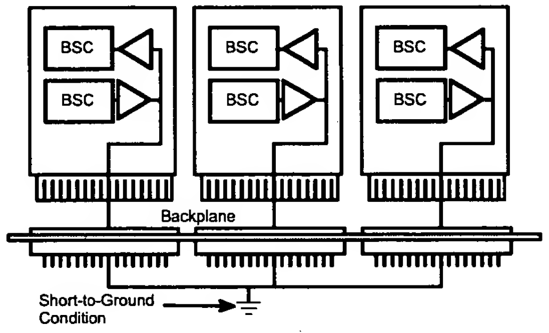


Figure 7-9. Backplane Short-to-Ground Fault

Bus contention faults occur when a three-state bus driver is stuck in a functional-enable mode. The BSCs in a controlled mode can be disabled by the test circuitry, thus bypassing the functional-enable control. All of the tests described can be executed offline in a test mode or by built-in test (BIT) software interfacing to an IEEE Std 1149.1-compliant serial scan controller in the system.

Summary

Board-to-board fault detection and isolation in a controlled offline test mode and concurrent background BIT execution can be dramatically improved by using boundary-scan testability devices within the bus interface circuitry of each board installed in a system. Boundary-scan architecture at the backplane-interface can increase fault isolation by masking some interface logic from the circuitry under test.

ASIC Verification

Very-high-density ASIC devices pose severe barriers to all testing, beginning with design verification and continuing with test and verification of the manufactured part.

Solution

IEEE Std 1149.1-based testing can be used for testing and verifying ASIC devices if the design process, beginning with design specifications, includes scan-based testing. This example cites the development of a *vector processor* ASIC with 165,000 gates.

Detailed Description

The test logic in the vector processor includes internal scan and built-in emulation logic, in addition to boundary scan and the optional *INTEST* capability. The emulation capability includes control of the processor, implementing RUN, STOP, SINGLE-STEP, and EXAMINE/MODIFY internal registers, and real-time breakpoints.

Test Development

Design verification, test hardware, and test software are developed concurrently with the ASIC. Verification code consists of two types of tests:

- Scan-based tests and diagnostics
- Embedded test code

A boundary-scan test tool is used to execute and monitor boundary-scan tests. The tests are used to verify test bus operation, scan path integrity, internal-register access, core-logic operation, and IC-to-IC interconnects. An expanding set of scan-based utilities and special-purpose test programs are written to perform specific tests or troubleshoot hardware problems. These utilities are developed from the time hardware integration is begun.

In addition to the scan-based tests, embedded test code is developed in microcode to verify instruction set execution, register and memory decode, memory access, and input/output operations. The embedded tests target functional logic and verify at-speed operations.

Design Verification

The vector processor ASIC in the design incorporates extensive internal scan paths and breakpoint logic controlled via the IEEE Std 1149.1 test bus. Key registers that are accessible via scan include the address and data register files, ALU and multiplier pipeline registers, and the program instruction register.

Initial checkout of the vector processor includes scan path integrity and internal register access using internal scan and boundary scan. The first devices contain nonfunctional microcode RAM, therefore an alternate plan is devised to use IEEE Std 1149.1 to validate all other circuitry.

The vector processor checkout uses the boundary-scan architecture to perform register reads and writes and simple operations to verify microcode execution. Simple microcode operations are tested by scanning an opcode into the instruction register and clocking the processor using boundary scan.

A test routine is developed in C++ that automates the microcode execution steps. The program reads microcode-object files, scans the instruction into the instruction register, clocks the processor, and reads the program counter to locate the next instruction. As each instruction is executed, the program dumps the accumulator and status registers to allow monitoring of each instruction. This process is repeated for each microcode routine until all microcode is executed and nearly all processor functions are verified using the "bad" vector processors. Many weeks are saved while waiting for new vector processors.

Summary

Design verification, hardware test, and software integration represent significant challenges. Anticipating this, many design-for-test and integration capabilities need to be incorporated into the device to ensure controllability and observability of embedded functions. IEEE Std 1149.1 provides the primary method for design verification, hardware/software integration, and hardware test.

Test and interaction issues in this case were aggravated by the lack of physical access caused by the small geometry of the module. Boundary-scan testing saved a significant portion of the design verification effort that would have been required for the less-effective conventional physical access methods.

Memory-Testing Techniques

The steady increase in memory density has led to the increased use of high-capacity memory boards in today's electronic systems. It has also encouraged the use of embedded memory within microprocessors, digital signal processors (DSP), and ASICs. As a result, testing of these boards and devices has become more complex. Large-capacity memory boards require very long test execution times for adequate fault detection. Large bus sizes (32 bits) and high bus fanout further complicate the problem of isolating component faults. Embedded memories, which often make up a significant portion of the device area, create additional difficulties due to a lack of test access.

Solution

The use of scan-path technologies can simplify the problem of testing large memory arrays within a system or embedded memories within a device. Scan-path devices such as TI's bus-interface components can be used to partition large memory device arrays into smaller, more easily testable arrays to increase component fault isolation. In addition, the use of TI's IEEE Std 1149.1-compliant macro cells within ASIC devices provides controllability and observability of embedded memory that is normally inaccessible. Memory test times can be reduced through the efficient partitioning and testing of several memory arrays (on a board or within a device) in parallel.

Detailed Description

Figure 7-10 shows a 256 × 8 memory array, associated bus interface, and control logic configured to test various memory-pattern-generation techniques. This configuration allows for explicit read/write operations using IEEE Std 1149.1 *EXTEST* and *SAMPLE* instructions. In addition, the components have a BIST capability (controlled by IEEE Std 1149.1 instructions) used to perform memory read/write operations. The test results shown in Table 7-1 compare explicit scanning in the RAM array address, data, and strobe signal to using an IEEE Std 1149.1-instruction-controlled BIST to generate address, data, and strobe signals automatically at the test clock (TCK) rate of 6.25 MHz.

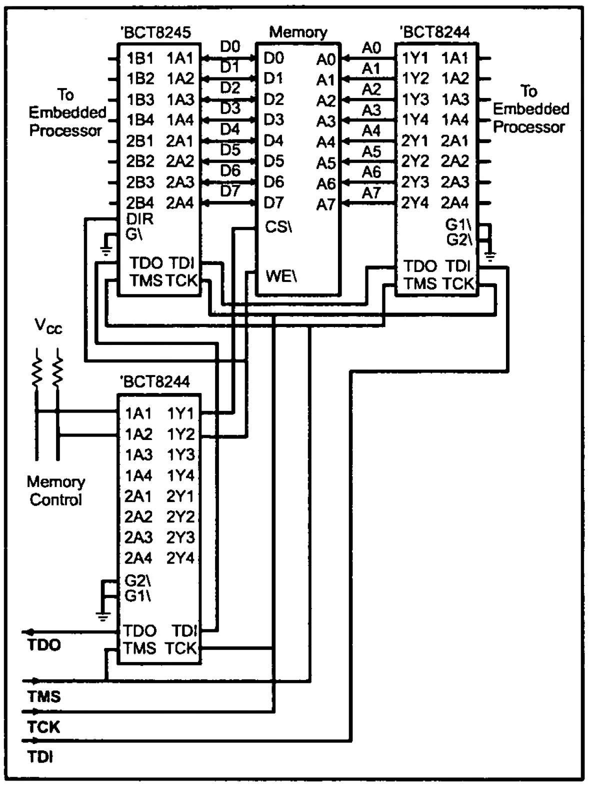


Figure 7-10. Block Diagram of Simplified Boundary-Scannable RAM Interface

Table 7-1. Access Rates of IEEE Std 1149.1 Devices With and Without BIST

Mode		**256 Accesses**	**1,000,000 Accesses**
IEEE Std 1149.1 (With BIST Capabilities)			
	Time to Apply	0.00041 seconds	<0.01 minutes
	Number of Scans	7	28,000
	Number of Patterns	512	2,000,000
IEEE Std 1149.1 (Using *EXTEST* and *SAMPLE*)			
	Time to Apply	5.625 seconds	375.00 minutes
	Number of Scans	512	2,000,000
	Number of Patterns	512	2,000,000

Table 7-1 shows that the BIST controlled by IEEE Std 1149.1 instructions overcomes the limitations of conventional memory testing. Using boundary-scan *EXTEST* and *SAMPLE/PRELOAD* solves the direct physical access problem, but it is time consuming. By using BIST circuitry embedded within the functional logic surrounding large memory arrays, IEEE Std 1149.1 tests can be designed to emulate closely the characteristics and timing speeds of the memory being accessed. TI's bus-interface components offer the IEEE Std 1149.1-controlled BIST functionality required to test memory, while providing the electrical signal conditioning and buffering a design engineer typically designs around a microprocessor.

In a similar fashion, TI's family of IEEE Std 1149.1-compatible cells can be used within an ASIC device to provide access to embedded memory normally inaccessible through external pins. The IEEE Std 1149.1-compatible cells support the same self-test features as the bus-interface devices (PRPG, PSA, Toggle 1/0 modes), which further simplify memory testing. Additional features such as programmable PRPG and PSA taps, built-in comparison logic, maskable PSA/comparison inputs, and bidirectional cells provide even more flexibility in self-test design. These features can be used with a built-in test controller to create an autonomous BIST for the memory array and/or other functions within the device.

The use of IEEE Std 1149.1 cells in performing read/write tests on RAM is shown in Figure 7-11. Address sequencing is performed using the PRPG mode. The PRPG and toggle modes automatically generate data patterns to be written to the memory cells under test while the memory write enable is controlled by direct scan. When reading from memory, data patterns are verified using the comparison logic in the IEEE Std 1149.1-compatible cells. This is possible since each IEEE Std 1149.1-compatible cell can latch and remember the last data pattern that was written when configured correctly. During memory reads, the latched pattern is fed back for comparison with the data being read. A pass/fail signal for the test can be generated by combining the comparison outputs of each IEEE Std 1149.1-compatible cell.

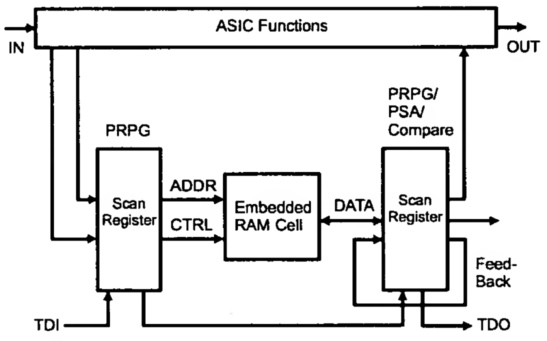


Figure 7-11. Testing Embedded RAM

A similar testing method can be used when testing read-only memory (ROM) embedded within a device, as shown in Figure 7-12. During ROM tests, address sequencing and read enables are controlled as in Figure 7-11, and the PSA mode is used to compress the data output into a representative signature. If desired, comparison logic can be used to compare the calculated signature with a known good signature to generate a memory pass/fail signal.

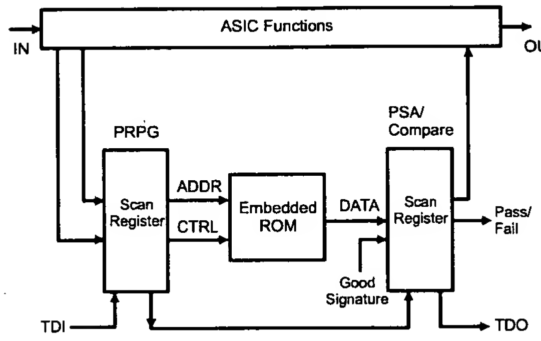


Figure 7-12. Testing Embedded ROM

Summary

The use of TI's bus-interface components and boundary-scan cells provides increased fault detection and isolation capabilities when used to partition large-capacity memory boards and devices with embedded memory. The increased partitioning allows the use of parallel testing and autonomous built-in self test, greatly reducing the overall test execution times normally attributed to memory testing. IEEE Std 1149.1 bus-interface components provide simple replacements for buffers and transceivers that are normally used in memory system design, while creating little impact on system performance. TI's boundary-cell library is a completely configurable set of test cells that allows the designer to create sophisticated test structures (such as built-in self test for memories), while requiring only a minimal knowledge of test engineering principles.

Backplane Multidrop Environment

The backplane environment, which carries many signals in parallel to all the boards, contains certain conditions not found in other test situations. A typical backplane with a complement of boards (such as a computer that uses different accessory boards) may be configured differently for different purposes. In other cases, certain boards may be capable of interacting independently of normal backplane activity, and some boards may include BIST. For maximum usefulness and short test times, a test program must accommodate a partially populated backplane, while allowing more than one board to be active at a time.

IEEE Std 1149.1 was developed to serially access ICs on a board, but it also can be used to access boards installed in a backplane. By making simple hardware modifications to the backplane circuitry, ring or star backplane configurations can be accessed using normal IEEE Std 1149.1 commands.

In a backplane configured as a ring (see Figure 7-13), all boards directly receive the TCK and TMS control outputs from a primary test-bus controller (TBC). The scan path is daisy chained between the TBC's TDO output and TDI input. During a test, the TBC drives TMS and TCK to scan data through all boards in the backplane, through the TDO/TDI bus connections. A serial path works only if all the boards are present and can scan data from their TDI input to TDO output. A missing or faulty board breaks the path, so the TBC cannot scan data through the backplane.

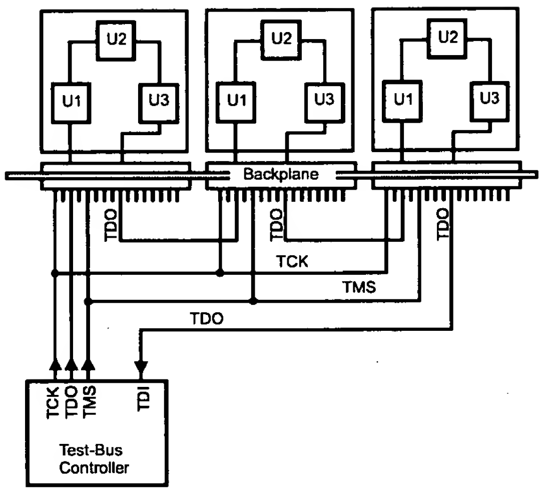


Figure 7-13. Backplane Ring Configuration

In a backplane star configuration (see Figure 7-14), the TBC drives TCK and TDI directly to all boards and each board outputs a TDO signal to the TBC. Each board requires a unique TMS signal from the TBC and a separate circuit trace for each signal. During testing, only one board is enabled at a time and only the TMS signal for that board is active. In a backplane with 50 boards, the TBC needs 50 individual TMS signals, with circuit traces for each. Another drawback is that only one board can be scanned at a time.

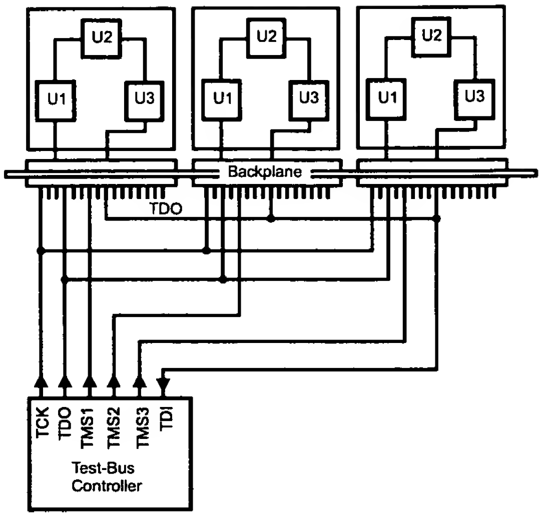


Figure 7-14. Backplane Star Configuration

Solution

A new serial bus protocol has been developed by Texas Instruments that is interpreted by a TI device called the Addressable Scan Port (ASP) (SN74ABT8996†). This device provides linkage between IEEE Std 1149.1-compatible boards and a backplane. The ASP becomes the IEEE Std 1149.1 input bus for each board in the backplane. It intercepts the protocol on the normal incoming IEEE Std 1149.1 signal wires, allowing it to connect or disconnect its board with the IEEE Std 1149.1 bus on the backplane. Once connected, the board can communicate using the IEEE Std 1149.1 protocol.

The ASP protocol is invoked when the IEEE Std 1149.1 bus is in one of the following four TAP states: Test-Logic-Reset, Run-Test/Idle, Pause-IR, or Pause-DR.

† *SN54ABT8996, SN54ABT8996 10-Bit Addressable Scan Ports Multidrop Addressable IEEE 1149.1 (JTAG) TAP Transceivers*, Literature Number SCBS489A, Texas Instruments 1996.

The protocol can address and select any ASP-equipped board in the backplane. After the board is connected to the IEEE Std 1149.1 bus, the ASP simply passes data through to the bus so normal operation can resume.

Detailed Description

Figure 7-15 shows a backplane with ASP devices on each board, allowing individual boards to be configured and active simultaneously.

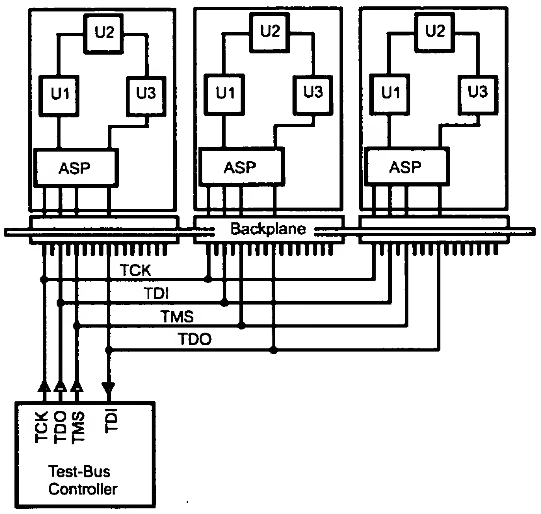


Figure 7-15. Backplane With ASP-Equipped Boards

For example, if the interconnects between two boards are to be tested, it is necessary to select and scan the first board to output a test pattern, then select the other board to receive the test pattern. The ASP makes it possible to select and scan the second board without resetting the first board. During operation, each pattern of the interconnect test is scanned out of the receiving board before a new pattern is scanned into the driving board.

Another test may be to select and initiate self-tests in a selected group of backplane boards. The ASP allows BIST to start on each board and run while other board tests are initialized. Each board is then polled to read the test results, one board at a time.

Summary

The special problems associated with backplane testing can be accommodated when the SN74ABT8996 Addressable Scan Port is added to individual boards.

Embedded Applications

Field testing can be simplified if equipment is configured to take advantage of embedded IEEE Std 1149.1 functions.

Solution

TI's SN74ACT8990† Test-Bus Controller (TBC) and SN74LVT8980‡ Embedded Test-Bus Controller (eTBC) devices can be embedded in system circuitry to provide autonomous IEEE Std 1149.1-based testing. A licensable, embeddable "C" source code product called Scan Engine, which includes test vector translators, is available to simplify embedded boundary-scan test applications.

Detailed Description

TI's test-bus controller devices operate under the control of a host processor to master one or more IEEE Std 1149.1 scan paths. Appearing to the host as memory- or I/O-mapped peripherals, they accept parallel command and test data and convert it to appropriate IEEE Std 1149.1 protocols, as required. Thus, they allow the host software to operate at a high-level (essentially, at the level of SVF) rather than requiring that it encode knowledge of TMS state, sequencing, etc.

† *SN54ACT8990, SN74ACT8990 Test-Bus Controllers*, Literature Number SCAS190C, Texas Instruments 1996.

‡ *SN54LVT8980, SN74LVT8980 Embedded Test-Bus Controllers*, Literature Number SCBS676, Texas Instruments 1996.

Host Interface

Both TBC and eTBC devices implement a generic host interface. Each contains a set of memory- or I/O-mappable registers that can be read/written by the host to control its operation and monitor its status.

The host interface of the TBC consists of a 16-bit data bus, 5-bit address bus, read strobe, and write strobe. The 24-register set includes two command registers, in addition to scan-data FIFOs, configuration registers, status registers, and counter registers.

The host interface of the eTBC consists of an 8-bit data bus, 3-bit address bus, read/write select and read/write strobe. The 8-register set includes one command register in addition to scan-data FIFOs, configuration registers, status register, and counter register.

IEEE Std 1149.1 Interface

The TBC implements six separate TMS output signals allowing it to directly master up to six separate star-configured scan paths. The TBC also has two separate TDI inputs that can be used to support scan paths with different requirements for buffering of their TDO signals (for example, scan path(s) that are on the same PCB as the TBC versus scan path(s), which are on remote PCBs).

The eTBC implements a single set of TAP-mastering outputs, allowing it to directly master only one scan path. However, the eTBC provides a wealth of features that provide exceptional flexibility in control of scan test operations:

- Programmable division of TCK output frequency allows support for slow-scan-rate components while still operating fast-scan-rate components at higher speed.
- Discrete mode of TAP control enables arbitrary TMS/TDO sequences to allow for support of non-compliant components or specialized test extensions.
- Gated mode of TCK output provides mechanism for directly supporting 'ABT8996 Addressable Scan Port (ASP) shadow protocols in multidrop configurations.

Command Set

Both TBC and eTBC devices support high-level command sets that allow the host to directly request IEEE Std 1149.1 state changes, scan operations, or periods of execution in the Run-Test/Idle state.

While the TBC can be made to generate ASP shadow protocols by proper sequencing of configuration changes and scan operations, the eTBC command set contains direct ASP scan commands.

Isolating the TBC From the IEEE Std 1149.1 Bus

Each of the TBC and eTBC devices provides a mechanism for allowing its IEEE Std 1149.1 interface to be disabled. When the TOFF* of the TBC is asserted or the OE* of the eTBC is released, the respective TAP-mastering outputs are forced to a high-impedance state. By so disabling the IEEE Std 1149.1 interface, an external controller (for example, a scan-based ATE) is allowed to take control over the embedded scan path.

Host Requirements

A host processor executing embedded test software (for example, Scan Engine) needs three primary hardware resources:

- Memory- (or I/O-) mapped parallel access to the TBC/eTBC
- ROM/EPROM space for the object code
- RAM or ROM space for the test vectors. These test vectors are broken into two primary blocks, stimulus data and response data.

It should be noted that the stimulus data could either be downloaded into RAM or be permanently programmed into ROM/EPROM.

Summary

The SN74ACT8990 Test-Bus Controller and SN74LVT8980 Embedded Test-Bus Controller devices offer the designer of high-reliability equipment a way to embed automatic IEEE Std 1149.1-compliant test circuitry into products. Scan Engine software provides an easy way to control the TBC in an embedded system. The solution discussed here maintains IEEE Std 1149.1-compatibility and access to test vectors developed during design verification. The result is continuous quality assurance with direct traceability to development data.

Boundary-Scan Test Flow

When new ASICs or boards come from the manufacturing line, the first concern is that the devices work as designed. High-density ASICs and multi-layer PWBs with surface mount devices cannot be adequately tested without boundary-scan testing.

Solution

Use boundary-scan techniques and tools on products designed with IEEE Std 1149.1-1990-compatible components.

Detailed Description

Figure 7-16 summarizes an efficient verification process using BSDL supplied by manufacturers of IEEE Std 1149.1-compatible ICs and HSDL developed for the ASICs and PWB.

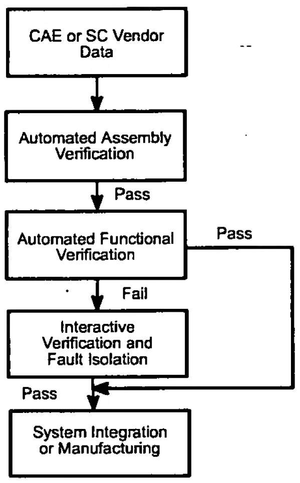

CAE or SC Vendor Data
Automated Assembly Verification
Pass
Automated Functional Verification
Pass
Fail
Interactive Verification and Fault Isolation
Pass
System Integration or Manufacturing

Figure 7-16. General Boundary-Scan Test

Automated Assembly Verification

Automated assembly verification depends on getting BSDL or HSDL descriptions for IEEE Std 1149.1-compatible devices from the manufacturer and developing corresponding HSDL for modules, boards, and systems. Figure 7-17 shows the flow that verifies a correct scan path and interconnections between boundary-scan cells.

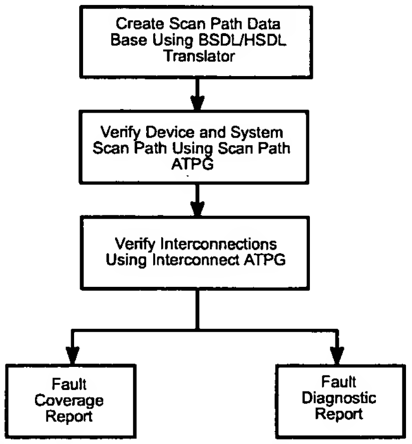

Create Scan Path Data Base Using BSDL/HSDL Translator
Verify Device and System Scan Path Using Scan Path ATPG
Verify Interconnections Using Interconnect ATPG
Fault Coverage Report
Fault Diagnostic Report

Figure 7-17. Assembly Verification Flow

Devices that pass assembly verification must then be tested for functionality.

Automated Functional Verification and Fault Detection

Functional verification begins with IC testing, followed by virtual component or cluster testing and system verification. These flows and the tools involved are shown in Figure 7-18.

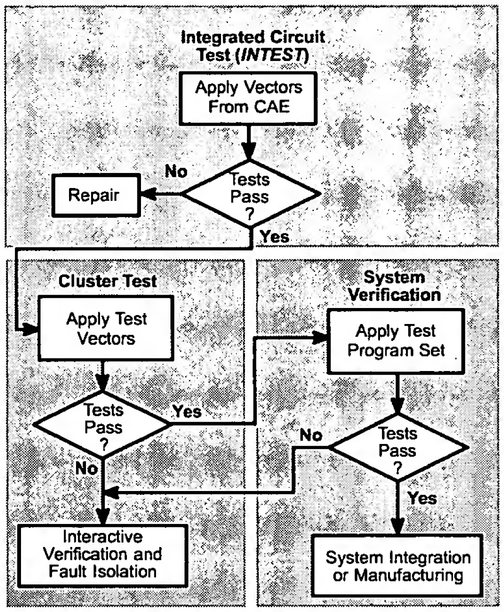


Figure 7-18. Automated Functional Verification and Fault Detection

No matter which flow is used, the goal is to reuse data whenever possible. For functional verification we want to reuse any serial or parallel vectors developed in the CAE system; those vectors become a pass/fail test. When CAE vectors are not available, boundary-scan test tools support test stimulus development approaches such as vector recording, macros, and high-level-language programs.

All functional tests highlight failures down to the vector level. Knowing which vector failed can reduce the time spent in interactive testing to isolate the problem to a device, register, or pin.

Interactive Verification and Fault Isolation

Figure 7-19 shows how faults in units failing previous tests can be isolated using scan-based test tools.

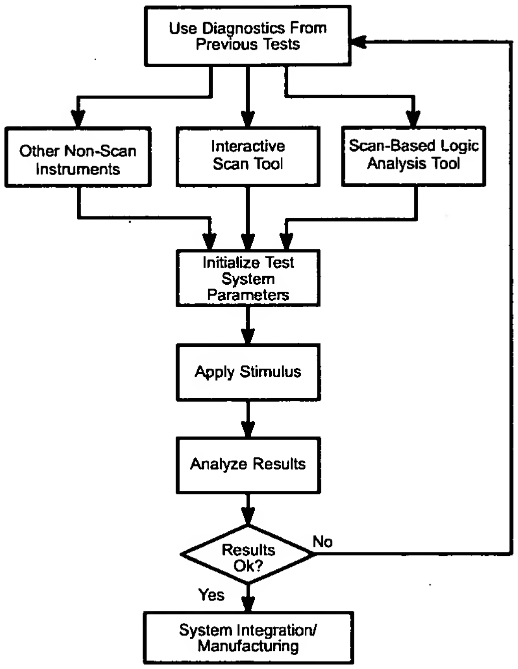

Use Diagnostics From Previous Tests
Other Non-Scan Instruments
Interactive Scan Tool
Scan-Based Logic Analysis Tool
Initialize Test System Parameters
Apply Stimulus
Analyze Results
Results Ok?
No
Yes
System Integration/ Manufacturing

Figure 7-19. Interactive Verification and Fault Isolation

Summary

Design verification and fault isolation processes are greatly enhanced when a full suite of boundary-scan tools and techniques are used.

Chapter 8
Product Summary

This chapter provides a brief description of TI's comprehensive offering of IEEE Std 1149.1-compatible components and a listing of TI's IEEE Std 1149.1 (JTAG) boundary-scan logic product family.

IEEE Std 1149.1-Compatible Components

TI has a long history of involvement with JTAG and IEEE Std 1149.1 and of support for integrated circuits that incorporate IEEE Std 1149.1-compatible boundary scan. Starting with release of the first commercially available boundary-scan logic ICs in May 1990, only 3 months after ratification of IEEE Std 1149.1, TI has lead the semiconductor industry in its offerings of products that support board- and system-level testability.

In addition to TI's broad line of boundary-scan logic devices (see below), which are implemented primarily for the purpose of building board- and system-level test architectures, TI supports IEEE Std 1149.1-compatible ICs across its broad product lines. From ASIC to DSP, networking to multimedia, TI's portfolio includes many boundary-scan devices.

Highlights

ASIC — To facilitate the implementation of IEEE Std 1149.1-compliant boundary scan in TI's ASIC products, boundary-scan-cell modules have been developed for CMOS standard-cell and gate-array libraries.

DSP — Select functions in TI's 'C4x, 'C5x, 'C54x, and 'C8x (MVP) generations incorporate IEEE Std 1149.1 boundary scan. In addition, all functions in these generations and in the 'C2xx generation use the test access port (TAP), as specified by IEEE Std 1149.1, to access on-chip emulation features.

Networking — Currently, several functions from ThunderLAN™ (Ethernet™) and ThunderCELL™ (Asynchronous Transfer Mode) families offer IEEE Std 1149.1 boundary scan:

IEEE Std 1149.1 (JTAG) Boundary-Scan Logic

TI's family of IEEE Std 1149.1 (JTAG) boundary-scan logic includes octal bus interface, Widebus™, and scan-support functions that incorporate circuitry that allows them, and the electronic systems or subsystems in which they are used, to be tested without reliance on traditional probing techniques.

In all, over 30 devices, composed of a wide selection of BCT and ABT octal bus interface, ABT(H) and LVTH Widebus™, and scan-support functions, are currently available.

For data sheets and documentation, see the *1997 Boundary-Scan Logic, IEEE Std 1149.1 (JTAG) 5-V and 3.3-V Bus-Interface and Scan-Support Products Data Book*, literature number SCTD002.

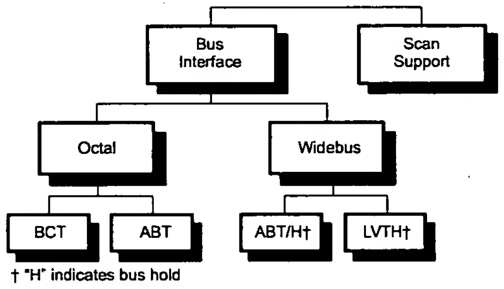


Figure 8–1. TI's Family of IEEE Std 1149.1 (JTAG) Boundary-Scan Logic

Bus Interface

Bus-interface functions are available in BCT, ABT, and LVT (3.3-V ABT) technologies, in octal and Widebus™ (18- and 20-bit) options of standard buffers, latches, flip-flops, transceivers, and registered transceivers. Featured is the universal bus transceiver (UBT™) which, in a single package, can serve all of these common bus-interface roles. In addition, ABT and LVT technologies offer devices with bus-hold inputs and integrated series-damping resistors. Package options for the bus-interface devices include plastic dual in-line (PDIP), small-outline integrated circuit (SOIC), shrink small-outline package (SSOP), thin shrink small-outline package (TSSOP), and plastic thin quad flat package (TQFP).

Octal Bus Interface

SN74BCT8240A	24	Octal Inverting Buffer/ Driver
SN74BCT8244A	24	Octal Buffer/ Driver
SN74BCT8245A	24	Octal Bus Transceiver
SN74BCT8373A	24	Octal Transparent D-Type Latch
SN74BCT8374A	24	Octal D-Type Flip-Flop
SN74ABT8245	24	Octal Bus Transceiver
SN74ABT8543	28	Octal Registered Bus Transceiver
SN74ABT8646	28	Octal Registered Bus Transceiver
SN74ABT8652	28	Octal Registered Bus Transceiver
SN74ABT8952	28	Octal Registered Bus Transceiver

Widebus™— TQFP

SN74ABT18502	64	18-Bit Universal Bus Transceiver
SN74ABT18504	64	20-Bit Universal Bus Transceiver
SN74ABT18646	64	18-Bit Registered Bus Transceiver
SN74ABT18652	64	18-Bit Registered Bus Transceiver
SN74ABTH18502A	64	18-Bit Universal Bus Transceiver
SN74ABTH18504A	64	20-Bit Universal Bus Transceiver
SN74ABTH18646A	64	18-Bit Registered Bus Transceiver
SN74ABTH18652A	64	18-Bit Registered Bus Transceiver
SN74LVT18502	64	18-Bit Universal Bus Transceiver
SN74LVTH18502A	64	18-Bit Universal Bus Transceiver
SN74LVTH18504A	64	20-Bit Universal Bus Transceiver
SN74LVTH18646A	64	18-Bit Registered Bus Transceiver
SN74LVTH18652A	64	18-Bit Registered Bus Transceiver

Widebus™— TQFP, with series-damping resistors

SN74ABTH182502A	64	18-Bit Universal Bus Transceiver
SN74ABTH182504A	64	20-Bit Universal Bus Transceiver
SN74ABTH182646A	64	18-Bit Registered Bus Transceiver
SN74ABTH182652A	64	18-Bit Registered Bus Transceiver
SN74LVT182502	64	18-Bit Universal Bus Transceiver
SN74LVTH182502A	64	18-Bit Universal Bus Transceiver
SN74LVTH182504A	64	20-Bit Universal Bus Transceiver
SN74LVTH182646A	64	18-Bit Registered Bus Transceiver
SN74LVTH182652A	64	18-Bit Registered Bus Transceiver

Widebus™ — TSSOP

SN74ABT18245A	56	18-Bit Bus Transceiver
SN74ABT18640	56	18-Bit Inverting Bus Transceiver
SN74LVTH18245	56	18-Bit Bus Transceiver
SN74LVTH18512	64	18-Bit Universal Bus Transceiver
SN74LVTH18514	64	20-Bit Universal Bus Transceiver
SN74LVTH18516	64	18-Bit Universal Bus Transceiver
SN74LVTH18640	56	18-Bit Inverting Bus Transceiver

Widebus™ — TSSOP, with series-damping resistors

SN74LVTH182245	56	18-Bit Bus Transceiver
SN74LVTH182512	64	18-Bit Universal Bus Transceiver
SN74LVTH182514	64	20-Bit Universal Bus Transceiver
SN74LVTH182516	64	18-Bit Universal Bus Transceiver
SN74LVTH182640	56	18-Bit Inverting Bus Transceiver

Universal Bus Transceiver (UBT™)

The UBT is the flagship of TI's IEEE Std 1149.1 bus-interface offerings. With its dual-ranked latch storage and generalized control logic, the basic UBT can perform all of the most common bus-interface roles: driver, latch, flip-flop, transceiver, latched transceiver, and registered transceiver. Enhanced UBT functions offer additional features such as clock enable and stored/real-time data multiplexers. With the added feature of boundary scan, TI's IEEE Std 1149.1 UBTs are the ideal solution for nearly all bus-interface needs.

Table 8-1. IEEE Std 1149.1 UBTs Replace 50+ Bus Functions

FUNCTION	8 BIT	9 BIT	10 BIT	16 BIT	18 BIT	20 BIT
Transceiver	'245,'623,'645	'863	'861	'16245,'16623	'16863	'16861
Buffer/ Driver	'241,'244,'541		'827	'16241,'16244, '16541	'16825	'16827
Latched Transceiver	'543			'16543	'16472	
Latch	'373,'573	'843	'841	'16373	'16843	'16841
Registered Transceiver	'646,'652			'16646, 16652	'16474	
Flip-Flop	'374,'574		'821	'16374		'16821
Standard UBT*					'16500, '16501	
Universal Bus Driver					'16835	
'18502 and '18512 UBTs Replace Above Functions						
Registered Transceiver w/CLK Enable	'2952			'16470,'16952		
Flip-Flop w/CLK Enable	'377	'823			'16823	'16721
Std UBT* w/CLK Enable					'16600, 16601	
'18516 UBT Replaces Above Functions						
'18504 and '18514 UBTs Replace All Above Functions						

*Non-Boundary-Scan Device

Scan Support

Scan-support functions facilitate implementation of board- and system-level test architectures. These include devices for mastering the IEEE Std 1149.1 test bus, adapting the IEEE Std 1149.1 test bus to multidrop configuration, performing at-speed functional test, and partitioning the scan path into smaller, more manageable segments.

SN74LVT8980	28	Embedded Test-Bus Controller
SN74ACT8990	44	Test-Bus Controller
SN74ABT8996	24	Addressable Scan Port
SN74ACT8994	28	Digital Bus Monitor
SN74ACT8997	28	Scan-Path Linker
SN74ACT8999	28	Scan-Path Selector

Test-Bus Controllers

Embedded Test-Bus Controller (eTBC) — The SN74LVT8980 provides a simple, off-the-shelf solution that can be hosted by a low-cost 8-bit microcontroller or microprocessor to generate the signals required to master the IEEE Std 1149.1 test bus. Its LVT implementation means that this device is the industry's first test-bus solution to operate at 3.3 V, while it still interfaces to scan-path devices that use 5-V power.

Test-Bus Controller (TBC) — The SN74ACT8990 provides a simple, off-the-shelf solution that can be hosted by a 16-bit microcontroller/microprocessor to generate the signals required to master the IEEE Std 1149.1 test bus.

Backplane/Multidrop Test-Bus Support

Addressable Scan Port (ASP) — IEEE Std 1149.1 defines how to implement boundary scan at the IC level but does not address board- or system-level architectures. With TI's SN74ABT8996 device, backplane or other multidrop applications now can be supported.

At-Speed Functional Test

Digital Bus Monitor (DBM) — The SN74ACT8994 is a special-function device that provides the capability to emulate the functions of logic and signal analyzers. The DBM can be embedded in systems as a non-intrusive at-speed monitor. A single DBM monitors up to 16 signals, but several can be used in concert for width or depth expansion.

Scan-Path Support

Scan-Path Linker (SPL) — The SN74ACT8997 provides for scan-path partitioning by allowing selection, under IEEE Std 1149.1 scan control, of any combination of up to four secondary scan paths. Multiple SPL devices can be cascaded to support any desired number of secondary scan paths, as well as multilevel hierarchies.

Scan-Path Selector (SPS) — The SN74ACT8999 provides for scan-path partitioning by allowing selection, under IEEE Std 1149.1 scan control, of any one of four secondary scan paths.

Note:

Not all devices mentioned are available at the time of this printing. Contact your TI sales representative or authorized distributor for commercial or military grade availability information.

Chapter 9
Other Support and Learning Products

This chapter provides information concerning learning tools available for IEEE Std 1149.1-1990.

Scan Educator

The Scan Educator PC-based tutorial software introduces the fundamentals of IEEE Std 1149.1, including architecture, protocols, and required instruction set. Self-paced and menu-driven, it contains both information and animated boundary-scan simulations. Hands-on exercises guide you through a boundary-scan test of an IEEE Std 1149.1-compliant octal at the bit-by-bit and register levels. You also are shown how to use boundary-scan testing with single or multiple devices for in-circuit observability and controllability.

Scan Board

No matter where you are on the boundary-scan learning curve, TI's Scan Board can help you move to higher levels of understanding — quickly and easily. Scan Board can be a learning tool for new users of boundary-scan and a test and evaluation platform for the experienced user. This specially developed board comes with multiple IEEE Std 1149.1-compliant devices that can help you evaluate both the tools and the concepts behind board-level testing. With Scan Board, you can perform boundary-scan interconnect tests and insert various faults and fault types, as well as demonstrate board-level built-in self-test (BIST) for testing memory, internal test (*INTEST*) of an ASIC, internal BIST of an ASIC, cluster testing, single- and multiple-scan paths, and many other capabilities. Scan Board comes with schematics, netlists, X-Y plots, data sheets, test patterns, BSDL descriptions, and a *free* executable software program. This software allows you to apply test patterns supplied with Scan Board, or to apply your own.

Scan Board demonstrates these advanced test techniques and more:

- Scan-path verification
- Interconnect testing
- Pins-in (IC core logic) testing
- BIST methods at the device and board level
- Boundary-scan interconnect automatic test-pattern generation (ATPG)
- Functional cluster logic tests using boundary-scan and BIST capabilities
- Fault detection and isolation
- Control and monitoring of discrete event lines
- Counting, toggle, and pseudorandom pattern generation (PRPG) stimulus methods
- Parallel signature analysis (PSA) response capture
- Scan-path switching

Flexibility

Scan Board incorporates basic boundary-scan features for new users to learn boundary-scan testing, and also incorporates advanced extensions such as BIST, *INTEST*, internal scan, and multiple scan paths for advanced users. Scan Board's physical layout separates independent multiple functions both electrically and physically, while maintaining a single continuous IEEE Std 1149.1 scan path for ease of test manipulation. Fault-injection switches are provided so the user can practice fault-isolation techniques. Scan Board provides for the injection of 12 different faults. Fault types include: stuck-at-one, stuck-at-zero, bridging, non-adjacent bridging, internal device, and scan path.

Compatibility

Scan Board is compatible with in-circuit test fixtures and testers. Nodes are accessible via boundary-scan or from the bottom with adequate and uniform spacing for in-circuit probes. Nets without both the boundary-scan driver and receiver are accessible via 40-mil probe points with 100-mil spacing.

Supporting Documentation Included

Scan Board comes with board description and operation, schematics and diagrams, X-Y and Gerber plots, a fault table, and boundary-scan device data sheets. Electronic files include BSDL, HSDL, Netlists (Viewlogic, EDIF 2.0.0, and Teradyne CDB), interconnect patterns (SVF), ASIC internal test patterns (SVF), and cluster-logic test patterns (SVF). Scan Board demonstration software is included at no extra charge.

CD-ROM

TI has combined the key elements of IEEE Std 1149.1 on a single CD-ROM. Now the latest versions of the IEEE Std 1149.1-1990 (including IEEE Std 1149.1a-1993) are merged into one searchable document.

Also included are TI's:

- Test-bus evaluation report
- Application notes
- Data sheets for TI boundary-scan products

The test-bus evaluation (TBE) report was written by TI's Defense Systems & Electronics Group to evaluate the applications and impact of standard test buses on overall system testability. The TBE report is based on years of test-bus development and applications experience at TI; since the early 1980s, TI has played a key role in defining test buses, first, for the VHSIC program, and then later as a key contributor in both JTAG and IEEE Std 1149.1.

This CD-ROM brings together current information on IEEE Std 1149.1. Microsoft and UNIX environments are supported.

Testability Video Tape

Texas Instruments offers a 74-minute, two-part video to explain the concept of testability. The tapes are available in U.S. (NTSC) and international (PAL) formats.

Tape 1 — The first tape explains the growing importance of testability and the tremendous potential impact on cost, time, and quality that design for test (DFT) can have. It also describes the work of the Joint Test Action Group (JTAG) and the provisions of IEEE Std 1149.1.

Tape 2 — The second tape examines the standard in more detail. It explains the test-bus architecture and instructions and shows how to use boundary-scan testing on a board or system. The tape includes software-support standards, example application, and information on how boundary-scan test tools help automate boundary-scan testing. The tape clearly explains the function of each component of the standard, and shows how they work together to provide an accurate, dependable test procedure.

Boundary-Scan Video Outline

- Introduction — Testing problems and concepts
- Economics — Benefits and cost-reduction potential
- Design-for-Testability Concept — Controls, diagnosis, DFT strategies, and more
- Introduction — IEEE Std 1149.1 Boundary Scan
- Details of IEEE Std 1149.1
- Testing With IEEE Std 1149.1 Boundary Scan — During design and development, production, field testing and maintenance. Also in this section: testing the boundary-scan core logic at the board level, optimized interconnect tests, isolating interconnect faults, testing boards with a mix of boundary-scan and nonboundary-scan components, testing mixed-signal boards, and exploiting BIST during tests.
- Digital Applications of IEEE Std 1149.1 Boundary Scan — Data formats supporting boundary scan, plus application notes

- Proposed Product Development and Support Cycle
- Commercially Available Products for DFT
- Available Training on DFT and Boundary Scan

Appendix A
Abbreviations/Acronyms

AC	Alternating Current (dynamic)
ASIC	Application-Specific Integrated Circuit
ASP	Addressable Scan Port
ASSET	Advanced Support System for Emulation and Test
ATE	Automatic Test Equipment
ATG	Automatic Test Generation
ATPG	Automatic Test-Pattern Generation
BIST	Built-In Self-Test
BIT	Built-In Test
B/S	Boundary Scan
BSC	Boundary-Scan Cell
BSDL	Boundary-Scan Description Language
BSR	Boundary-Scan Register
BST	Boundary-Scan Test
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
DBM	Digital Bus Monitor
DC	Direct Current (static)
DFT	Design for Test
DIP	Dual In-Line Package
DR	Data Register
DSP	Digital Signal Processor
EDA	Electronic Design Automation
ESD	Electrostatic Discharge
ETBC	Embedded Test-Bus Controller
FIFO	First-In, First-Out (memory)
HDL	Hardware Description Language
HSDL	Hierarchical Scan Description Language
IC	Integrated Circuit
ICE	In-Circuit Emulation
ICT	In-Circuit Test
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output

IR	Instruction Register
ISP	In-System Programming
JEDEC	Joint Electronic Device Engineering Council
JIT	Just-In-Time (manufacturing)
JTAG	Joint Test Action Group
LSB	Least-Significant Bit
LSSD	Level-Sensitive Scan Design
MSB	Most-Significant Bit
PCB	Printed Circuit Board
PPM	Parts Per Million
PRPG	Pseudorandom Pattern Generation
PSA	Parallel Signature Analysis
PWB	Printed Wiring Board
SPL	Scan-Path Linker
SPS	Scan-Path Selector
SSA	Serial Signature Analysis
SVF	Serial Vector Format
TAP	Test Access Port
TBC	Test-Bus Controller
TCK	Test Clock
TDI	Test Data Input
TDL	Test Description Language
TDO	Test Data Output
TMS	Test Mode Select
TPS	Test Program Set
UBT	Universal Bus Transceiver
UUT	Unit Under Test
VHDL	VHSIC Hardware Description Language
VHSIC	Very High-Speed Integrated Circuit
XCVR	Transceiver

Appendix B
Glossary

Automatic Test Generation
A functional test pattern generation that is done algorithmically on a computer

Boundary-Scan Description Language (BSDL)
A hardware description language developed to describe devices built to IEEE Std 1149.1. See also HSDL.

Built-In Self Test
Logic included within a design that can apply test signals and compare results to determine if the design is working correctly

Concurrent Fault Simulation
A fault simulation method that simulates only fault models that differ from the fault-free model at the node being simulated. The node being simulated is an event-driven node.

Controllability
The ability to set a logical value on a signal node within the design. Controllability can be a measure of how difficult it is to generate test patterns. If a signal is hard to control, it is difficult to set up values necessary to test design features.

Design for Test
A method that incorporates rules and techniques in the design process to make testing easier

Detectable Fault
A possible failure that has at least one test pattern or sequence of patterns that, when applied to a device, detect failure

Deterministic Test Vectors
A fixed-order set of test patterns

Fault
A design failure or flaw that causes an incorrect response to input stimuli

Fault Grade
The percentage of modeled faults that will be detected by a test sequence when applied to a device. The fault grade is determined through fault simulation.

Fault Model
A simulator representation of a fault to model an unexpected behavior of a failure

Fault Simulation
The process of simulating test patterns using fault models and a fault-free model to determine whether potential failures will be detected with the test sequence

Gate-Pin Fault
A fault modeled at the connection of a gate pin to the signal-node net

Hierarchical Scan Description Language (HSDL)
A superset of BSDL developed by TI. HSDL is used to describe IEEE Std 1149.1-compliant devices, modules, and systems. See also BSDL.

Initialization
The process of placing a circuit or unit under test into a known state for testing or logic simulation

Interconnect Testing
A test process that verifies correct connections between ICs and modules on a board

Level-Sensitive Scan Design (LSSD)
The IBM DFT methodology of scan design. "Level Sensitive" means that all registers are activated with voltage levels rather than clock edges.

Linear Feedback Shift Register (LFSR)
A shift register counter that can be used for pseudorandom test pattern generation and for compression of test pattern results

Load Fault
A fault model that is a load on a wired junction

Logical Redundancy
Design logic that, removed from the design, performs and provides the same static logic responses. Logical redundancy may be purposefully added to improve speed or drive performance.

Modular Testing
Testing a design by applying independent test sequences to test isolatable modules. Each module has its own test sequence that is not affected by other modules in the design. With modular testing, ATG and fault grading can be performed on a single module at a time.

Multiple Fault
A fault model created by simultaneously representing the fault with multiple stuck-at faults

Node Fault
A fault model that ties an interconnection net to a fixed logic level. A node fault is equivalent to all sources on a net being source faults with the same logical value.

Observability
The capability to observe the effect of a node value at an output pin. If the node value changes, the output value also changes.

Parallel Fault Simulation
A method that simulates a set of fault models at the same time as the fault-free model

Parallel Signature Analyzer
An LFSR used to compress test data by collecting data from the design and exclusive ORing the resulting test data with register-state data

Parametric Test
A test that checks the voltage, current, and timing specifications of a design. The ac and dc propagation tests are parametric tests.

Propagation
The response of an output of a gate (or net) to an input of the gate (or net)

Scan-Path Design
A DFT methodology where all or a selected group of storage elements can be serially interconnected to facilitate testing of embedded combinational logic in an integrated circuit

Scan-Path Linker (SPL)
TI's SN74ACT8997 device acts as a scan-path multiplexer, allowing dynamic path switching. The SPL can select any combination of four secondary scan paths.

Scan-Path Selector (SPS)
TI's SN74ACT8999 device acts as a scan-path multiplexer, allowing dynamic path switching. The SPS can select one of four secondary scan paths at a time.

Scan Testing
The DFT technique that connects all internal registers in a design into a shift register used during test. The scan shift registers, called scan paths, are used to load any desirable state before application of a test over combinational logic between serial (scannable) register latches and unload any resultant state for observation.

Serial Vector Format (SVF)
A standard ASCII-based pattern format for expressing IEEE Std 1149.1 scan operations and state transitions. This vector format was jointly developed by TI and Teradyne to allow pattern transportability between test tools.

Source Fault
A fault model that is a source to a wired-junction net. Gate outputs are examples of source faults.

Stuck Closed (Shorted Gate) Fault
A stuck-at fault model in which the transistor controlled by the gate is always closed. This has the same effect as shorted data lines. This fault model is the same as an SA1 fault on the gate input of an n-p-n transistor or an SA0 fault on the gate input of a p-n-p transistor.

Stuck Fault
A failure model in which a gate pin is modeled as stuck to VCC (for stuck-at-one, SA1) or to GND (for stuck-at-zero, SA0)

Stuck Open (Open Gate) Fault
A stuck-fault model in which the transistor controlled by the gate is always open. This fault model is the same as an SA0 fault on the gate input of an n-p-n transistor or an SA1 fault on the gate input of a p-n-p transistor.

Test Access Port (TAP)
Port described in IEEE Std 1149.1-1990. Must contain a minimum of three input connections (TDI, TMS, TCK) and one output connection (TDO). A fourth input connection (TRST*) is optional. The TAP provides access to test functions built into a component.

Test Description Language (TDL)
A test pattern format that describes the stimuli and expected results for a test application. TDL tests are applied using I/O connector pins.

Test Generation
The process of producing a test sequence to test a design. Three methods of test generation are: automatic test generation (ATG), manual test preparation, and random input values generation.

Test Pattern
A set of input stimuli values that are applied at the beginning of a period of time and a set of output observation values made at the end of a period of time that are represented by a string of values

Test Pattern Grading
The process of fault simulating a test pattern sequence to obtain a fault grade

Test Point Insertion
The DFT technique that includes additional connections to internal nodes for either control or state observation

Test Vector
Another name for a test pattern

Testability
The property of a circuit, board, or system that enables testing

Undetectable Fault
A fault that cannot be detected by any achievable test pattern. Undetectability may be due to logic redundancy or to logic that cannot be observed.

Unpredictable Fault
A fault that causes the value of a node to be indeterminate, thereby making it impossible to detect a fault

Appendix C
References

1. Agrawal, Vishwani D., "Sampling Techniques for Determining Fault Coverage in LSI Circuits," *Journal of Digital Systems*, Vol. V., Number 3, pp. 189-203.

2. Bennetts, R. G., *Introduction to Digital Board Testing*, Crane, Russak & Company, Inc., NY, NY, 1982.

3. Bhavsar, D. K., and R. W. Heckelman, "Self-Testing by Polynomial Division," *Digest of IEEE Test Conferences*,1981, pp. 208-216.

4. Bleeker, van den Eijnden, de Jong, *Boundary-Scan Test — A Practical Approach*, ISBN 0-792-9296-5.

5. Frohwerk, R. A., "Signature Analysis: A New Digital Field Service Method," *Hewlett-Packard Journal*, May 1977, pp. 2-8.

6. IEEE Std 1149.1-1990 (includes IEEE Std 1149.1a-1993), *IEEE Standard Test Access Port and Boundary-Scan Architecture*, ISBN 1-55937-350-4.

7. IEEE Std 1149.1b-1994, *Supplement to IEEE Standard Test Access Port and Boundary-Scan Architecture*, ISBN 1-55937-497-7.

8. Jandhyala, S. and A.W. Ley, "*Design-for-Test Analysis of a Buffered SDRAM DIMM*," 1996 Proceedings, International Workshop on Memory Technology, Design, and Testing.

9. Ley, A. W., "*A Look at Boundary Scan from a Designer's Perspective*," 1994 Proceedings, Electronic Design, Automation & Test Asia Conference.

10. Maunder, Colin M. and Rodham E. Tulloss, *The Test Access Port and Boundary-Scan Architecture,* ISBN 0-8186-9070-4.

11. Parker, Kenneth P., *The Boundary-Scan Handbook*, ISBN 0-7923-9270-1.

12. Peterson, W. W., and E. J. Weldon Jr., *Error Correcting Codes*, MIT Press, Cambridge, MA, 1972.

13. Pynn, C., *Strategies for Electronics Test*, McGraw-Hill, Inc., NY, 1986.

14. Sridhar, T., D. S. Ho, T. J. Powell, and S. M. Thatte, "Analysis and Simulation of Parallel Signature Analyzers," 1982 International Test Conference, Philadelphia, PA, November 15-18,1982, pp. 656-661.

15. Thatte, S. M., D. S. Ho, H. T. Yuan, T. Sridhar, and T. J. Powell, "An Architecture for Testable VLSI Processors," 1982 International Test Conference, Philadelphia, PA, November 15-18, 1982, pp. 484-493.

16. Turner, M.E., D. G. Leet, Ronald J. Prilik, and D. J. McLean, "Testing CMOS VLSI: Tools, Concepts, and Experimental Results," 1985 International Test Conference, Philadelphia, PA, November 19-21, 1985, pp. 322-328.

17. Whetsel, L., "A Proposed Standard Test Bus and Boundary Scan Architecture," 1988 IEEE International Conference on Computer Design, Rye Brook, NY, October 3-5, 1988, pp. 330-333.

18. Whetsel, L., "A Proposed Method of Accessing 1149.1 in Backplane Environment," 1992 Proceedings, International Test Conference.

19. Whetsel, L., "*Hierarchically Accessing 1149.1 Applications in a System Environment*," 1993 Proceedings, International Test Conference.

20. Williams, T. W., "VLSI Testing," *Computers*, Vol. 17, No. 10, October 1984, pp. 126-136.

Appendix D
Internet Starting Points

Institute of Electrical and Electronics Engineers (IEEE)
http://www.ieee.org/index.html

IEEE Std 1149.1
http://www.computer.org/tab/tttc/standard/s1149-1/home.html

Test Technology Technical Committee (sponsor for IEEE 1149.x standards)
http://www.computer.org/tab/tttc/

Texas Instruments
http://www.ti.com

Texas Instruments Semiconductor Group
http://www.ti.com/sc/docs/schome.htm

Texas Instruments Advanced System Logic
http://www.ti.com/sc/docs/asl/home.htm

Texas Instruments IEEE 1149.1 (JTAG) Boundary Scan
http://www.ti.com/sc/docs/jtag/jtaghome.htm

Texas Instruments IEEE 1149.1 Silicon Product and BSDL Availability
http://www.ti.com/sc/docs/jtag/silicon.htm

Texas Instruments Product Information Data Sheet Search
http://www.ti.com/sc/docs/psheets/pids.htm

Index